AFRL-RI-RS-TR-2008-196
Final Technical Report
July 2008

A COGNITIVE FRAMEWORK FOR RESOURCE-AWARE SENSOR NET ORGANIZATIONS

University of Massachusetts

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2008-196 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
LOK KWONG YAN Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUL 2008	Final	Jan 05 – Dec 07

4. TITLE AND SUBTITLE

A COGNITIVE FRAMEWORK FOR RESOURCE-AWARE SENSOR NET ORGANIZATIONS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-05-1-0039

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Daniel D. Corkill

5d. PROJECT NUMBER
NBGQ

5e. TASK NUMBER
10

5f. WORK UNIT NUMBER
08

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Massachusetts
140 Governors Drive, CMPSCI Dept.
Amherst MA 01003-9264

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RITB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2008-196

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 08-3892

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes research associated with CNAS (Cognitive Network for Atmospheric Sensing), an agent-based, power-aware sensor network for ground-level atmospheric monitoring. To conserve battery power, CNAS sensor agents must have their WiFi radios turned off most of the time. Research activities were performed in five main areas: implementation of a blackboard-based agent architecture that could be effective given the limited computing facilities at each sensor agent; development of the CNAS agent software and atmospheric sensing knowledge sources; improving CNAS performance and responsiveness with limited radio availability; power-aware reasoning associated with solar harvesting; and exploration of potential next-generation CNAS hardware. This report discusses the issues addressed, the techniques developed, evaluations and successful field deployments of CNAS, and lessons learned. The report concludes with a summary of remaining technical challenges and recommendations for future research and development activities.

15. SUBJECT TERMS
Power-aware, agent-based sensor networks; ground-level atmospheric monitoring; weather; solar harvesting; cognitive computing; blackboard systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lok Kwong Yan
U	U	U	UU	60	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Contents

1 Introduction

CNAS[1] (Collaborative Network for Atmospheric Sensing) is an experimental, agent-based, power-aware sensor network for ground-level atmospheric monitoring [1]. CNAS is a research and demonstration tool developed jointly by AFRL/RI (Rome, NY) and The University of Massachuetts Amherst (UMass) for conducting realistic explorations of the advantages and limitations of agent-based environmental monitoring. The CNAS effort is investigating the use of hardware capabilities that are likely to become cost-effective for production deployments in the next few years.

A combination of blackboard and multi-agent system (MAS) techniques are used in CNAS sensor agents. Blackboard systems [2, 3] are proficient in supporting indirect and anonymous collaboration among software entities and in exploiting temporal decoupling of entity interactions in order to obtain maximum flexibility in coordinating activities. MAS researchers, on the other hand, have developed effective techniques for operating in highly distributed, dynamic settings and for coordinating local, autonomous activity decisions. These capabilities are all highly valuable assets in developing an effective software architecture for agile, resource-aware sensor network agents.

The UMass portion of this research focused on research activities in five main areas:

- implementation of a blackboard-based, individual agent architecture that could be effective given the limited computing facilities at each sensor agent
- development of the CNAS agent and network software architecture and atmospheric sensing knowledge sources (KSs)
- improving CNAS performance and responsiveness with limited radio availability
- power-aware reasoning associated with solar harvesting performed by each sensor agent
- exploration of potential next-generation CNAS hardware

This final report discusses the issues we addressed, the techniques we developed, evaluations and demonstrations of CNAS, and lessons learned. Much of our research as well as details of CNAS deployments appeared in four publications, which are included as appendices to this report. The report concludes with a summary of remaining technical challenges and recommendations for future research and development activities.

2 Areas of Research

Each of the main areas of research performed in conjunction with this effort is summarized in the following sections.

2.1 Deploying a Blackboard-System on each Sensor Agent

One objective of the CNAS effort was demonstrating the feasibility of hosting a high-level language and an Artificial Intelligence (AI) blackboard-system framework on the PASTA (Power-Aware Sensing, Tracking, and Analysis) microsensor platform[2] [4] developed by the University of Southern California's Information Sciences Institute (ISI). As envisioned by the AFRL/RI team, the PASTA microsensor with its Intel PXA255-based ARM[3] CPU would form

[1] Pronounced "See-nas." The original CNAS acronym was short for Cognitive Network for Atmospheric Sensing, but "Cognitive" has since evolved into "Collaborative," emphasizing the importance of the agent-based interactions supporting the *cognitive* reasoning activities in CNAS.

[2] http://pasta.east.isi.edu/

[3] Previously called the Advanced RISC Machine and originally named the Acorn RISC Machine.

the core of each CNAS sensor agent.

In addition to the PASTA, each CNAS sensor node would also be equipped with a Crossbow MTS420CA sensor board providing:

- Intersema MS5534AM barometric pressure sensor
- TAOS TSL2550D ambient light sensor
- Sensirion SHT11 relative humidity/temperature sensor
- Leadtek GPS-9546 GPS module (SiRFstar IIe/LP chipset)
- Analog Devices ADXL202JE dual-axis accelerometer[4]

A Netgear MA111 wireless adapter connected to the PASTA's Universal Serial Bus (USB) interface would provide standard IEEE[5] 802.11b wireless communication in the 1–2km range required for CNAS. This hardware, and a 12V battery, would be packaged in a PVC housing to position the wireless antenna and sensors 4.25′ above ground level.

After a preliminary assessment, we felt that it was indeed possible to support both Common Lisp and the GBBopen[6] open-source blackboard-system framework (see page 3) on the PASTA, and we initiated a porting effort. GBBopen is written in Common Lisp and uses CLOS (the Common Lisp Object System) [5] and the Common Lisp Object System Metaobject Protocol (MOP) [6] to provide blackboard-specific object capabilities. The blending of GBBopen with Common Lisp transfers all the advantages of a rich, dynamic, reflective, and extensible programming language to blackboard-application developers. Thus, GBBopen's "programming language" includes all of Common Lisp in addition to the blackboard-system extensions provided by GBBopen.

We initially pursued a partially completed ARM port of SBCL (Steel Bank Common Lisp)[7] as the Common Lisp implementation for the PASTA. SBCL is an open-source Common Lisp implementation that supports an excellent optimizing native-code compiler. Although SBCL's compiler technology supports both RISC and Intel-class processors, the combination of a RISC instruction set with a relatively limited number of registers (more Intel-like) on the ARM processor did not match any of the existing compilation models in the SBCL compiler. This would require implementing a new "hybrid" compilation model for the ARM processor, and we learned that this additional effort had delayed the volunteer work required to finish the ARM port of SBCL indefinitely.

Since this contract did not provide the time or resources to invest in completing SBCL's ARM port, we were forced to abandon the SBCL strategy. Fortunately, as a backup, we had also been exploring the possibility of porting the GBBopen blackboard-system framework to CLISP (http://clisp.cons.org/), another open-source Common Lisp implementation that uses a small C-based kernel in conjunction with a platform-independent bytecode compiler and virtual-machine executer. When we first looked at CLISP, it did not support sufficient MOP capabilities for GBBopen. As luck would have it, however, just as SBCL became infeasiable, sufficient MOP support for GBBopen was completed for CLISP. CLISP was already ported to ARM processors, so with the added MOP support that became available in CLISP 2.34, we had a viable Common Lisp implementation for hosting GBBopen on the PASTA.

An important advantage of CLISP is the small memory footprint that results from its small C-based kernel and byte-code representation. A major disadvantage of CLISP, however, is

[4]Not used in CNAS.

[5]The IEEE name was originally an acronym for the Institute of Electrical and Electronics Engineers, Inc. Today, the organization's scope of interest has expanded into so many related fields, that it is simply referred to by the letters I-E-E-E (pronounced "Eye-triple-E").

[6]http://GBBopen.org/

[7]http://sbcl.sourceforge.net/

that it does not support Lisp multiprocessing (process threads), which complicates real-time event processing. In hindsight, the space-over-execution-speed savings of CLISP's byte-code representation was a good choice for our CNAS sensor agents, leaving plenty (30MB) of memory space on the PASTA for atmospheric-sensing application code and data at each sensor agent.

We were able to make use of the Debian ARM packaging of CLISP 2.34 (performed by Will Newton). The Debian package allowed us to bypass cross-compiling the CLISP kernel that would have required using an ARM cross-compilation toolchain running on a Linux/x86 host and then bootstrapping the rest of the build directly on the PASTA (which would have been a very painful process). Using the Debian CLISP package did introduce some problems, however. Differences between Debian and the PASTA's TinyOS Linux distribution required forced bypassing of package dependencies and introduced incompatibilities in several shared libraries. The latter resulted in some degradation in CLISP's memory management and garbage-collection performance on the PASTA.

GBBopen

CNAS sensor agents, console nodes, and regional nodes are implemented using the open-source GBBopen framework. GBBopen is a modern, high-performance, open source blackboard-system development environment that is based on concepts that were explored and refined in the UMass Generic Blackboard system [7] and the commercial GBB product [8]. A major capability provided by GBBopen is multidimensional abstraction over blackboard objects ("unit instances"), blackboard levels ("space instances"), and proximity-based retrieval patterns. Multidimensional abstraction provides a semantically meaningful separation of blackboard-repository storage mechanisms KS and control code. This separation allows storage and search strategies and optimizations to change dynamically in order to maintain top performance. GBBopen also provides "link based" inter-object relationships and highly efficient and extensible event signaling/handling that form the foundation for fast, yet flexible, opportunistic processing activities and control reasoning.

GBBopen is written in the Common Lisp language, an ANSI (American National Standards Institute) standard.[8] Common Lisp is supported by commercial vendors and open-source implementations on a wide range of hardware platforms and operating systems. At the implementation level, GBBopen is designed as a smooth extension of Common Lisp, CLOS (the Common Lisp Object System) [5], and the Meta-object Protocol [6], providing all the advantages of a rich, dynamic, reflective, and extensible language to blackboard-application architects and component writers. These capabilities are crucial in building complex blackboard-based applications where representations, KSs, and control mechanisms will change during the development and over the operational lifetime of the application.

GBBopen incorporates over 25 years of experience in developing and delivering a wide range of blackboard-system-based applications. In addition to CNAS, GBBopen is used in many complex and challenging applications including these defense-related systems:

- CIFAR[9] (U.S. Army)—real-time information fusion of large-volume battlespace observations
- INCOMMANDS[10] (DRDC/DND Canada)—anti-missile threat assessment and weapon as-

[8]Common Lisp was the first ANSI standard to incorporate object-oriented programming: ANSI X3.226-1994.

[9]Collaborative Information-Fusion Assistant Reasoner

[10]Innovative Naval COMbat MANagement Decision Support TDP (Technology Demonstration Program) developed by the Defence Research and Development Canada (DRDC) arm of the Department of National Defence (DND) Canada.

signment for Halifax class Canadian Navy frigates

- COORDINATORs[11] (CMU (Carnegie-Mellon University) Team)—agent-based coordination support for humans

The PASTA running basic Linux system processes and an `ssh` remote-login session has approximately 34.3MB of free memory space (out of a total of 64MB). CLISP consumes slightly more than 2MB, and GBBopen uses another 2MB. This brings the free memory down to about 30MB that is available for blackboard objects, KSs, and sensor data—a reasonable amount for performing sensor-agent processing.

An Aside: Mobile CNAS agents

An investigation incorporating sensor-mobility decision making in CNAS was initiated early in this effort. Use of taskable mobile sensor agents greatly complicates resource-aware coordination. For example, can a sensor agent improve its ability to sense the environment or communicate by changing its location slightly?[12] Can communication be restored between disconnected subnetworks by repositioning one or more agents to bridge the transmission-range gap? Similarly, is improving coverage quality by moving sensor agents feasible given current power reserves?

The AFRL/RI team had procured and were experimenting with several Sony AIBO® "dog" platforms as low-cost mobility robots that could potentially carry a PASTA agent on their backs. At UMass, we purchased two Sony AIBOs, configured identically to those being used at AFRL, and Michael O'Neil, a graduate research assistant on the project, developed a set of command sequences for controlling an AIBO robot remotely from a PASTA sensor agent over Wi-Fi.[13]

One issue in using the AIBO was estimating the new location of the AIBO that results from a sequence of movement activities. We did not want to attempt to use AIBO-based sensing (such as video imagery) for locational feedback in this effort, so we began exploring the accuracy and repeatability of dead-reckoning movement estimation on flat, unobstructed surfaces. We felt that, should dead-reckoning not be sufficiently accurate for our needs, we could use GPS positioning on the sensor agent as a fall-back strategy. GPS (Global Positioning System) positioning is energy intensive, so we wanted to avoid relying on GPS positioning if possible.

In the fourth quarter of the effort, we completed our AIBO remote-command work. Further work on mobility planning was stopped, as we were re-tasked to focus all our resources on power-aware reasoning in the context of the planned PATRIOT 2006 demonstration exercise (page 7). The PATRIOT 2006 Exercise did not require mobile-agent capabilities.

2.2 CNAS Architecture

Two types of sensor agents are used in CNAS:

- **Sensor Agents:** A "basic" sensor agent consists of the PASTA and Crossbow hardware configuration described earlier.
- **TACMET-Augmented Sensor Agents:** A TACMET-augmented sensor agent is a basic CNAS sensor agent that also includes a Climatronics TACMET II 102254 weather sensor.[14]

[11]The "Coordination Decision Support Assistants" Program at DARPA's (Defense Advanced Research Projects Agency) Information Processing Techniques Office (IPTO).

[12]This possibility is at the heart of the recent DARPA LANdroids project.

[13]Wi-Fi is the trade name for the popular wireless technology used in most personal computers. The official position of the Wi-Fi Alliance is that Wi-Fi is merely a brand name that stands for nothing in particular, and they discourage the use of the term "Wireless Fidelity" that was originally used to market the branding.

[14]For readability, referred to simply as "TACMET" in this final report.

The TACMET provides a temperature sensor, a fast-response, capacitive relative humidity sensor, a barometric pressure sensor, a flux gate compass, and a folded-path, low-power sonic anemometer. Wind speed, wind direction (resolved to magnetic North using the flux gate compass), temperature, and relative humidity readings are provided to the PASTA over an RS-232C serial connection once every second. Unlike the Crossbow, TACMET measurements have been certified by the Air Force.

Each CNAS sensor agent performs basic sensing activities, obtaining atmospheric readings once each second. We concentrated initially on getting TACMET-augmented sensor agents operational. We developed a Common Lisp interface function to read the RS-232C serial connection once every second. Because CLISP is not multithreaded, we used a cooperative "polling function" mechanism integrated with GBBopen that ensured that the serial connection was read every second.[15] We also developed a TACMET data-stream representation where a list of report vectors was kept in decreasing temporal order. Each report vector contained 6 values: sensed time (seconds), temperature (degree C), relative humidity, pressure (millibars), wind direction (degrees from North), and wind speed (meters/second). The inverse temporal ordering allowed each new report (one per second) to be pushed efficiently onto the front of the list and the KS that is responsible for processing the inputs to chop off the tail when it was done processing reports.

Following Air Force meteorological practice, the following summary readings are computed from the 1-second readings and saved every five minutes:

- temperature (5-minute average)
- dew point (5-minute average)
- pressure (last reading)
- altimeter (last reading)
- wind-u-component (2-minute average, TACMET agents)
- wind-v-component (2-minute average, TACMET agents)

Sensor agents perform saturation-vapor-pressure, humidity-to-dew-point, pressure-to-altimeter, millibars-to-inches, and wind-meter-to-knot computations as needed.[16] All unsent 5-minute summary observations are transmitted during the next communication window to the sensor agent that is acting as the *cluster head* (discussed shortly).

In addition to the 5-minute summary observations, each sensor node performs an interval-based compression of the raw sensor observations. These compressed 1-second readings and the 5-minute summary observations are held by the agent for a user-specified period (typically for many days).

We also developed "dead reckoning" power-use models, using nominal figures for initial battery energy and consumption rates associated with various activities. Each sensor agent would keep track of the time spent in various states of power expenditure (radio on, GPS on, etc.) and deduct expenditures from the initial battery-energy figure. The dead-reckoning energy value was used as the estimate of the remaining power at the agent, as the PASTA does not provide any support for power monitoring and there were no sensors on the unmanaged battery.

[15] If the RS-232C interface is not read for several seconds, the serial port would lock up when its small character buffer became full. Thus, it was very important that the TACMET data was read promptly. A KS that required more than a small fraction of a second to execute needed to yield to the polling mechanism periodically to ensure that the port was read if a second had elapsed since the last reading.

[16] The PASTA does not include floating-point hardware, so these computations are performed by software emulation.

The wireless adapter used at each sensor agent is the component with the largest power-expenditure rate, by far. The adapter draws at a rate of 250mA when powered on, which would consume all battery power, operating alone, in 48 hours. Since we were constrained to use 802.11b in CNAS, the only solution was to have each agent turn off its wireless adapter most of the time. So, although the power to the central processor and peripheral modules at each sensor agent can be controlled independently, such power savings are secondary to the savings achieved by having the wireless adapter turned off as much as possible. We developed a shell interface to `/etc/init.d/wlan stop` that operates under KS control to power down the adapter (and `/etc/init.d/wlan start` to bring it back up again).[17] This fully turns off the USB wireless adapter, greatly reducing power expenditure. An agent's radio-power decisions cannot be made unilaterally, as other nodes need to know when their transmissions will be receivable, and nodes may be acting as forwarders in multi-hop transmissions that don't pertain to them.

Communication policies

We developed a set of compatible "time rendezvous" policies for determining when CNAS agents should collectively activate (and deactivate) wireless capabilities. These policies allow agents that may not be aware of the current policy to eventually synchronize their policy with others. Each policy consists of fixed-length communication windows that occur at regular intervals, where the windows of each policy align with one another whenever possible. The policies that we used in the CNAS are:

- **hourly:** A communication window occurs at the top of each hour
- **half-hourly:** A communication window occurs every 30 minutes, starting at the top of each hour
- **quarter-hourly:** A communication window occurs every 15 minutes, starting at the top of each hour
- **hourly-overnight-sleep** The hourly policy, but without communication after 6PM until 6AM (local time)
- **half-hourly-overnight-sleep** The half-hourly policy, but without communication after 6PM until 6AM (local time)
- **quarter-hourly-overnight-sleep** The quarter-hourly policy, but without communication after 6PM until 6AM (local time)

The current policy can be switched at the next communication window, based on current weather trends and mission objectives. A new or rebooted agent that is not aware of the current policy can be assured of communicating with others during the next daytime top-of-the-hour window, no matter which policy is in effect. Alternatively, the agent can be more aggressive and try connecting at the next quarter-hour window, and at subsequent fallback windows.

These policies assume that the clocks of all agents and the regional node are synchronized. Ensuring that the clocks at all agents are synchronized within a few seconds of one another was quite reasonable, given the GPS-based time signals that we planned to obtain from the Crossbow MTS420CA sensors at each sensor agent.

Inter-agent communication in CNAS uses standard TCP (Transmission Control Protocol) operating over an OLSR (Optimized Link State Routing)[18] multi-hop protocol. (Use of OLSR

[17] We actually have to take down the device using `/sbin/ifconfig wlan0 down` and pause several seconds before stopping the device.

[18] http://olsr.org/

was an externally imposed requirement for the CNAS effort.) OLSR is intended for dynamic routing under changing connectivity and propagation conditions, and where a relatively small proportion of nodes are likely to come and go at the same time. Because the radio has been powered off, OLSR re-initializes at the start of each communication window. Given this initialization requirement, each CNAS communication window was structured as the following sequence of activities:

0–60 sec	Wi-Fi power on, OLSR stabilization
60–120 sec	agent assessment, status exchange, high-priority message delivery, cluster head determination
120–140 sec	observation transmission (to cluster head)
140–240 sec	cluster head processing, low-priority message delivery
240–300 sec	cluster observation transmission (to console & regional nodes)
300 sec	Wi-Fi shutdown

The durations of these staged activity intervals are very conservative and provide substantial slack time for OLSR re-initialization and for coping with highly degraded communication. During a communication window, each agent uses an application-level message retransmission strategy whenever the TCP/IP-layer reports delivery failure. A prioritized store-and-retry message delivery strategy holds outgoing messages that cannot be delivered due to outage during a communication window as well as messages generated locally when the agent's radio is turned off.

Cluster heads

In addition to its sensor duties, a basic or TACMET-augmented sensor agent can also assume the role of cluster head. A *cluster* in CNAS is a grouping of sensor agents located in a geographic region of interest. Non-overlapping cluster regions are user-defined, and each agent determines its cluster membership at start up, based upon its location and the specified cluster regions.

We designed a cluster-head election strategy that is based on a total preference ordering of sensor agents in the cluster. Globally established criteria is used by each agent to compute an identical total preference ordering for assuming the cluster-head role over all the agents in its cluster. During the initial phase of every communication window, each sensor agent determines the agent that is the most preferred cluster-head among all the agents that appear to be alive in its cluster. If a sensor agent is not assuming the cluster-head role, it transmits its 5-minute observations to the cluster head. If it is the cluster head, it accumulates the observations received from the other agents in its cluster, creating 5-minute cluster summary observations.[19] As the end of the communication window draws near (at the 4-minute mark in our conservative policy), each cluster head transmits the summaries of its cluster to the regional node.

The cluster-leader determination strategy is very robust. For example, should a cluster become bifurcated, separate cluster heads will be selected for each cluster fragment. When connectivity is re-established, these cluster heads provide summaries for the same cluster to the regional node, where they can be combined into a single cluster summary. Furthermore, the most preferred sensor agent will again become the sole head of the reunited cluster.

PATRIOT 2006

CNAS was to be field tested at the 2006 PATRIOT Exercise to be held at Fort McCoy, Wisconsin in July 2006. Our research and implementation efforts shifted to support this demonstration.

[19]Cluster summaries include the list of the individual agents that contributed to them.

We developed a "command console" interface for the regional node that would allow a user to change the communication policy of the network, place the network in "debugging mode" (where the wireless adapters are not turned off), inspect data and evaluate expressions on any individual sensor agent, evaluate expressions on all sensor agents, obtain instantaneous sensor readings from any sensor agent, and so on. We also implemented a server facility for the regional node that would allow authorized users to connect to the regional node from any host on the internet and have their own CNAS "command console" available to them. We also implemented a METAR[20] publication facility that allowed the regional node to provide standard METAR summaries for each cluster.

To allow last-minute and even in-the-field software changes to CNAS, we developed a live-updating facility for CNAS sensor agents. This facility allowed new or updated software to be distributed from the regional node to all CNAS agents during the next communication window. Taking advantage of the dynamic nature of Common Lisp, these updates are compiled and integrated directly in each running agent. We developed a strategy where we could develop and test any such updates on a two sensor-agent "mini" CNAS network located in Amherst, Massachusetts. Tested updates could then be transmitted to the regional node at the PATRIOT 2006 Exercise via a telephone-line connection that was to be available on-site.

Field deployment of prototype systems seem to always generate surprises and unexpected issues, and the PATRIOT 2006 deployment was no exception (see the ATSN-07 article [1] reproduced in Appendix C). A major problem that surfaced before the Exercise involved firmware issues with the PASTA interface to the Crossbow MTS420CA sensor board. We worked with Joe Czarnaski at ISI to identify the problem, but kernel and firmware upgrades and even replacement PASTA stack boards did not provide reliable communication between the PASTA and the Crossbow. As a result, a decision was made to deploy CNAS at the PATRIOT 2006 Exercise with the Crossbow sensors disabled. Without the Crossbow, however, GPS positioning and system-clock setting was lost. To compensate, the AFRL/RI team decided to use a handheld GPS unit to determine the location of each sensor agent as it was placed. This location and the agent name (IP address) would then be entered into a table maintained by the CNAS regional node. Then, as each sensor agent came on-line, it would connect to the regional node in order to obtain its location.

The PASTA does not have a hardware clock, and the system clock has to be set every time the PASTA is booted. Without the Crossbow GPS-obtained time, we elected to have the regional node serve as a CNAS time server, and each sensor agent would synchronize its clocks to the regional node when it is booted. This would require a spreading-area strategy for sensor-agent deployment where the regional node would be operating and agents would be booted in a sequence where they would be able to contact the regional node in order to obtain the time (and their location information) either directly or via hops through already deployed sensor agents.

Even with the many issues described in Appendix C and last-minute changes in the way CNAS agents would operate with the Crossbow sensors and GPS disabled, the PATRIOT deployment was a success. Sensor agents adapted to communication and node failures, reassigned cluster-head roles as needed, and provided local atmospheric data as designed. Based on its performance at the PATRIOT Exercise, CNAS was selected for demonstration at the

[20]METAR is an aviation routine weather report. A METAR weather report is predominantly used by pilots as part of a pre-flight weather briefing, and by meteorologists, who use aggregated METAR information to assist in weather forecasting. The term METAR originated from the French phrase "message d'observation météorologique régulière pour l'aviation," a contraction of the French words MÉTéorologique (Weather) Aviation Reǵulière (Routine).

Talisman-Saber Combined Exercise conducted May–July 2007 in Queensland, Australia.[21]

Post-PATRIOT activities

With the PATRIOT 2006 Exercise concluded, we began removing the temporary Common-Lisp-based time setting code and regional-node-based positioning code that were added to CNAS at the last minute in support of the PATRIOT Exercise (where sensor agents had to operate without the Crossbow MTS420CA sensor board). We also integrated the patches that we had developed and used during the Exercise, and we added new code from our "to do" list that had been placed on hold as the PATRIOT 2006 deployment approached.

We replaced the boot-time CNAS start-up scripts used at each sensor agent with a service based script. Running CNAS as a service is more in keeping with other Linux system services and allows the CNAS software operating at a sensor agent to be restarted (even remotely) without rebooting.

In anticipation of a resolution to the Crossbow firmware issue, we changed CNAS from the regional-node-centric approach that we were forced to use at PATRIOT 2006 to our original sensor-agent-centered design that required GPS positioning and time-setting using the Crossbow. This original design allowed CNAS agents to self-configure and begin operations without relying on the regional node being present (an advantage in terms of both network resilience and lower communication during start-up). The updated CNAS design was tested using simulated Crossbow-GPS data while the hardware and firmware issues were being addressed by ISI.

We then extended the CNAS design to support a fourth type of node. In addition to basic sensor agents, TACMET-augmented sensor agents, and the regional node, we now supported *console* nodes. A console node can perform the same activities as the regional node, but console nodes do not have a connection to the Internet. A console node is a laptop or handheld computer that, upon entry into the CNAS-network area, can obtain observation data from the network. As with the regional node (and unlike sensor agents) console nodes do not turn their wireless radios on and off. An authorized user at a console node can change network objectives and policies, transition the network into debugging (ongoing communication) mode, and perform detailed inspection of the data and activities at individual sensor agents. Of course, unless the network is in debugging mode, these activities can only be achieved during communication windows when sensor agents have their radios turned on. During radio silence, commands are held at the regional or console node until communication with sensor agents becomes available.

We also implemented regional-node support for a Java-based graphical display client for visualizing CNAS data. Carrie Kindler (of ITT Industries) supported the viewer software. To minimize coding on the Java side, we used the Java program's existing directory-polling mechanism rather than a more direct socket-based service. The CNAS regional node creates XML files in a prearranged local directory which is scanned by the Java output viewer. We then added hourly sensor-agent-observation reporting to the regional node to support the Java-based graphical display client. We also implemented a sensor-agent-observation simulation capability and added it to the regional node to facilitate development and debugging of the Java display client with only the regional node running (without any "real" CNAS sensor agents operating).

[21] Talisman-Saber is a biennial series of joint exercises aimed at further developing and enhancing the defense relationships between the United States and Australia. It is the largest and highest priority Tier II exercise in the Pacific Theater.

Talisman-Saber 2007

CNAS was to be deployed at two different locations during the Talisman-Saber Exercise:

- aerial drop zone, a remote open area similar to the PATRIOT 2006 drop zone deployment
- Urban Operations Training Facility (UOTF), an urban environment constructed at the Shoalwater Bay Training Area

In order to support providing ground-level weather information to Air Force Weather Agency (AFWA) and Australian Bureau of Meteorology (BOM) weather servers, we modified CNAS reporting and METAR formatting to match Talisman-Saber Exercise requirements.

An important goal for the Talisman-Saber deployment was to have the Crossbow fully operational. Unfortunately, ISI was unable to resolve the PASTA firmware issue (see Section 2.5), and we were forced, once again, to operate without Crossbow sensors. We redesigned portions of the latest sensor-agent-centric protocols (again at the last minute) to accommodate sans Crossbow deployments at Talisman-Saber Exercise.

As with PATRIOT 2006, there were a few deployment surprises and challenges at Talisman-Saber. (see the upcoming ATSN-08 article [9] in Appendix D). Once again, however, the CNAS deployments met all the technical goals and objectives established for the Exercise. These included: 1) automatic dissemination and posting of weather observations to AFWA and BOM weather servers, 2) support of the AFRL "COUNTER" small UAV demonstration, 3) support of airdrop operations, and 4) adapting to changing user requirements, observation needs, and mission objectives. CNAS was deployed and providing information within 15 minutes of arrival at the drop-zone site—a highly visible achievement. As a result, CNAS has been invited to participate in the next Talisman-Saber Combined Exercise in 2009.

2.3 Improving Performance and Responsiveness with Limited Radio Availability

Collecting sensor readings, aggregating them together at the cluster heads, and communicating them to console or regional nodes is well suited to the periodic communication windows used in CNAS. However, when more dynamic activities need to be performed (such as inspecting the local state of an agent, retasking an agent's activities or changing network policies, or having agents modify the world around them), having to wait until the next communication window can be an issue (as well as frustrating). Techniques are needed that improve network responsiveness without increasing the total amount of time that agents' radios need to be turned on.

Routing improvements

During this effort, graduate student Huzaifa Zafar worked on a synthesis project investigating using organization knowledge of CNAS to improve routing performance. MASs benefit greatly from an organization design [10, 11] that guides agents in determining when to communicate, how often, with whom, with what priority, and so on. However, this same organization knowledge is not utilized by the general-purpose wireless network routing algorithms normally used to support agent communication. By making the routing algorithm aware of how agents interact in the CNAS organization, it can direct path exploration where it is most beneficial, find optimal paths faster, and conserve significant bandwidth for application use. General-purpose routing algorithms also treat all message as having the same priority. They spend the same amount of energy exploring paths for application messages where it is extremely important to find the best possible path as they do exploring paths for messages that are not as important and can be delayed. We wanted to make use of the organizational importance (priority) of various agent communications in controlling path exploration.

Huzaifa modified a proactive routing protocol (CQRouting [12]) to incorporate knowledge of the CNAS organization (which agents communicate with which other agents given their roles and cluster membership, and the priority of the messages being sent). Using a detailed network-level simulation of CNAS, we observed a large reduction in the number of exploration messages relative to traditional routing protocols (such as OLSR). This resulted in a significant (43%) increase in the communication bandwidth available to the CNAS application. We also obtained an increased global utility with fewer exploration messages by having the eCQRouting routing protocol bias path exploration based on message priority characteristics of the CNAS organization. This work is detailed in the upcoming AAMAS-08 paper [13] (see Appendix E).

Rapid radio cycling

We also began an investigation into using frequent, yet very brief, communication windows. Shorter communication windows would allow more of them (a higher communication-cycle frequency) to be supported with the same power expenditure. After Talisman-Saber, Zenon Pryk at AFRL/RI explored maintaining OLSR routing information across communication cycles. Experiments were run (using a small CNAS agent network deployed indoors at AFRL) where a small utility exercised all possible source and destination pairs. In this noisy indoor setting, OLSR stabilized in less than 15 seconds into the communication window versus the 40–50 seconds required by a from-scratch reinitialization. In a sparse, outdoor CNAS setting we expect the routing changes to be less dynamic due to the small number of paths available, and that OLSR with persistent routing tables will stabilize even faster.

We conjectured that the change in routing that might occur during the time that network radios were turned off would decrease as the duration of the off period decreased. This implied that it could be reasonable to begin communicating immediately upon powering on the wireless adapter, letting any OLSR re-stabilization occur as if a "normal" dynamic event had just occurred in a formerly stable network. With the OLSR re-stabilization period eliminated, each agent could turn on its radio and, as quickly as possible, determine if it needs to be available to support any message communication during that window.

This preliminary work is discussed further in the "Future Work" section (page 14).

2.4 Power-Aware Reasoning and Solar Harvesting

Sensor agents whose power reserves can be expected to be replenished from time to time add new challenges in power management. In conjunction with the AFRL/RI team, we selected the PowerFilm R15-600 roll-able (thin film on plastic) solar panel for CNAS sensor agents. The PowerFilm is a quality engineered and relatively affordable flexible panel that would be suitable for extended outdoor deployment.[22] A PowerFilm panel at each sensor agent would allow the agent's battery reserves to grow (up to full battery capacity) if the agent is in an unshaded location and the sun is shining. The activity decisions that are made by agents with replenishable resources now must consider how much additional power may become available and when. We assume that the CNAS sensor network is provided with the sunshine forecast (from sources outside the network) for the upcoming 24–48 hours, but that each agent needs to learn the implications of the forecast on its own power reserves. A sensor agent that recognizes that it is shaded by a hill in the afternoon should realize that it cannot expect to obtain much power replacement if the forecast is for cloudy weather until noon. On the other hand, an unshaded agent could anticipate being able to perform additional power-intensive activities, based on its expectation of power replenishment.

[22]The PowerFilm is a civilian version of a military panel.

In September 2006, Walter Koziarz (AFRL/RI) verified experimentally that the seventh field output by the "late-model" TACMET sensors is the battery voltage available at the TACMET. This finding addressed a major implementation requirement for our planned solar-visibility learning effort: knowing when the solar panel is generating power. We still needed to investigate the sensitivity of this voltage input reading to understand if it could be used to detect fairly instantaneous charging levels or if it could only be used to provide a course indication that the battery charge had changed after many minutes of charging.

A requirement of CNAS is that the positioning of sensor agents cannot be optimized for either data collection or solar harvesting. Eventually sensor agents may be airdropped (rather than hastily deployed by hand). In either case, the exact placement of sensors cannot be regulated, which requires the shade and tilt attenuation to be determined once the agents are situated. Thus, agents in CNAS, like agents deployed in other real-world applications, need to determine aspects of their local environments in order to make appropriate decisions. For predicting solar-harvesting amounts, each agent must develop a model of the efficiency of its solar panel, attenuation due to positioning of the panel relative to the sun, attenuation due to shading (by trees, hills, buildings, etc.), and attenuation due to cloud cover. With such a model, an agent can translate a temporal sunshine forecast into a realistic estimate for the solar energy to be harvested at its location. This model development must be done quickly, as power-management decisions will be inaccurate until each agent becomes aware of its surroundings.

We developed a strategy of factoring solar-harvesting model development (which would typically be done using multi-agent learning) into two phases: pre-deployment (site independent, individual agent) learning and post-deployment (site dependent, multi-agent) model completion. By performing as much site-independent learning as possible prior to deployment, we simplify what needs to be determined on-site by each sensor agent. Furthermore, we use collective sharing of local-observation information, in conjunction with temporal and spatial constraints in relating this information, to reduce the number of observations needed to perform each agent's model-completion activities. In all but the most unlikely of environmental conditions, our two-phased strategy allows individual-agent harvesting models to be completed using only the first and second day's observations. Thus each agent can make fully informed solar-harvesting predictions given cloud forecasts starting on the third day. This work is detailed in an upcoming ATSN-08 article [14] (see Appendix F).

2.5 Potential Next-Generation CNAS Hardware

The PASTA microsensor platform used in CNAS is already showing its age. New PASTA "Ziti" processors are no longer available and, as discussed above, problems interfacing the Crossbow to the PASTA remain unresolved. Similarly, the Netgear MA111 (IEEE 802.11b) USB Wireless adapter used in CNAS is outdated. After participating in the Talisman-Saber Exercise, we began looking into new hardware possibilities for CNAS agents.

One promising line of replacement processors are the PXA255-based Connex and the PXA270-based Verdex motherboards from Gumstix, Inc. These have become very popular, and they can be interfaced with compatible sensor, Wi-Fi, and GPS hardware. The Gumstix have the advantage of being commodity devices readily available at commodity prices. However, they have the disadvantage of not being designed or targeted for low-energy consumption. The PASTA has a distributed-peer architecture that can decouple processing from peripheral operation, allowing even the central processor to be powered down to lower total system-power expenditure [4]. For use as a CNAS agent, however, this limitation is not a serious liability. Managing the high-expenditure Wi-Fi and GPS devices has been the prime focus of

our CNAS effort, and having the Gumstix running continually is acceptable.

Porting CLISP to the Gumstix

On the PASTA, we were able to make use of the Debian ARM packaging of CLISP 2.34 performed by Will Newton. The Debian package allowed us to bypass cross compiling the CLISP kernel. This was not the case for the Gumstix, as the kernel and library code there are tightly coupled with the buildroot cross-compilation toolchain. CLISP has a fairly complicated build process that has evolved over the years to support deployment on a wide range of hardware and operating system platforms. At one point, cross compilation of CLISP was supported, but that capability has long passed into disrepair. Attempting to use an ARM-based GNU development environment on the Gumstix itself was infeasible, so we had no alternative but to explore CLISP cross compilation.

Building CLISP involves the following eight steps:

1. `make init` — prepares all symbolic links and utilities
2. `make allc` — makes all `*.c` files
3. `make lisp.run` — makes the C-kernel executable
4. `make interpreted.mem` — creates a memory image with everything uncompiled
5. `make halfcompiled.mem` — creates a memory image with `compiler.fas` and the rest uncompiled
6. `make lispinit.mem` — makes all `*.fas` files and creates a memory image with everything compiled
7. `make modular` — makes the base module set
8. `make install` — completes the CLISP installation

These steps are normally executed in a fully automated fashion by a build "metascript." Due to cross-compilation, however, we were forced to perform them manually.

Working with contractor Steven Hoffman, we began the tedious process of deconstructing the automated build process for CLISP and manually debugging, adapting, and testing each step using the Linux/x86 buildroot cross-compilation toolchain followed by testing on the Gumstix. The key step in this process is creating the `lisp.run` CLISP kernel.

We developed a procedure for performing the first three steps (with manual interventions in the automated build scripts), to create a `lisp.run` executable for the Gumstix using the Linux/x86 buildroot cross-compilation toolkit. Compilation of the Common Lisp source files for the CLISP compiler and the rest of the CLISP Common Lisp environment was performed using a normal CLISP image running on the Linux/x86 development platform with appropriate compilation settings for the ARM-based deployment.[23] Then the compiled output files (`*.fas` files) and the `lisp.run` executable are copied to the Gumstix and the `lispinit.mem` memory image is built there. Again, this requires manual intervention, however the result is the latest CLISP release running smoothly on the Gumstix.

Given the suitability of CLISP for small, embeddable devices such as the Gumstix, restoring cross-compilation capability to the automated build scripts would eliminate the labor-intensive manual build process that we were forced to use. As an open-source project with a large user base, non-trivial changes to the current build process will require care and perseverance in

[23]With the exception of calls to some low-level routines, CLISP's byte-code compilation is platform independent. The compiler settings address these low-level issues.

order to be accepted. So, at least for now, recompiling CLISP for new system-software updates (such as the newly announced Gumstix OpenEmbedded (OE) build process) remains tedious.[24]

3 Lessons Learned

In this section, we reflect briefly on some of the broad lessons learned in performing this research effort. One obvious finding is that, when working with real-world, embedded-system hardware and wireless communication, "Whatever can go wrong, will go wrong" (and often even more will go wrong than that!). The specific challenges that were faced with CNAS and how they were met during field deployments are described in Appendices C and D.

Another important lesson from this effort involves the importance of addressing "real-world" engineering and conceptual challenges present in the actual application rather than conducting research based on intuition and assumptions. Instead of hindering our research, these real-world constraints forced us to reexamine past practices and techniques, and the result was a number of new approaches and research directions. For example, sensor agents operating near the limit of radio-communication range with their radios turned off most of the time present significantly different challenges than traditional MAS settings.

This effort also demonstrated the advantages of using a high-level language and an AI blackboard-system framework in CNAS. The flexibility of Common Lisp and GBBopen allowed us to rapidly develop and deploy significant changes to CNAS during research explorations and even in-the-field. PASTA processors running Linux, Common Lisp, and GBBopen provide a level of programming expressivity and on-the-fly modification that greatly facilitates the implementation of advanced behaviors and opportunistic-control decision making. We are fortunate in CNAS to have sufficient processing power and memory available to make shoe-horning complex behaviors and reasoning into tightly constrained C (and assembly) code a distant memory.

4 Remaining Technical Issues & Recommendations for Future Work

In this section, we describe several important directions for further research and empirical exploration.

Rapid radio cycling

Maintaining routing information across CNAS communication cycles and providing organizational communication information to the routing algorithms allow application-level communication to be performed using shorter communication windows. Shorter windows allow more of them (a higher communication-cycle frequency) to be supported with the same power expenditure. At the extreme, we would like to have each agent turn on its radio and, as quickly as possible, determine if it needs to be available to support any message communication during that window. If no communication support is needed (either to send a message, receive a message, or perform multi-hop forwarding), then the agent can turn off its radio straight away. The shorter these "turn on and determine" windows can be made, the more frequently they can be cycled while maintaining the same level of energy expenditure. Frequent cycles mean more CNAS responsiveness.

Intuitively, we could estimate the time needed for any agent to get an "I need to communicate" broadcast message to all other agents in the network. Then, when each agent begins

[24]The next CLISP release (2.45) is reported to address the cross-compilation support issue, and we are hopeful that fully automated cross-compilation CLISP builds will become available.

a cycle, it keeps its radio turned on for that long to hear if any agent (including console and regional nodes) wants to communicate. If no such message is received during that time, it turns off its radio.

For a small network, this broadcast time will be short, but the time increases as the number of hops grows. However, for very brief "no message" communication windows, the time until the next window is also short, so the need to communicate a message across the network in one cycle is lessened. If the message does not make it all the way to its destination in the current window, it will be stored and forwarded in the next. Thus, what we really seek is having the agents along a segment of the message path determine that they are needed in the current cycle. The goal is obtaining "waves" of communication-forwarding windows appearing where and when needed in the network. As with the organizationally based routing work described previously, CNAS can benefit by having the network-routing layer be aware of the application-level communication window and cycles. Making the "no message" decision at a lower communication layer can be very efficient.

As this contract ended, we were ready to begin development and evaluation of such a rapid-cycle, communication-window-aware routing protocol for CNAS-like networks. What differentiates this work from recent power-aware routing research is that we assume all agents' radios remain off (except for the frequent "any communication?" windows) until packets along a path need to be propagated. Rapid cycling is ideal for conserving communication energy in low-bandwidth, highly responsive settings, and we believe that it is an objective worth pursuing.

Next-generation hardware

Although we started exploring the use of Gumstix processors in the last months of this contract, we only began to consider possible sensor hardware for the Gumstix platforms. Given the growing success of this marketplace, however, the CNAS goal of exploring hardware capabilities that are likely to become cost-effective for production deployments in the next few years is quickly becoming a reality. Further tracking and exploration of commodity hardware and sensor developments will remain important.

Alternate communication technologies

In this effort, CNAS agents were required to communicate using a "stock" OLSR routing protocol operating on IEEE 802.11b Wi-Fi devices. Following the Talisman-Saber Exercise, this program-objective restriction was relaxed, allowing exploration of improvements to standard OLSR and consideration of alternative routing approaches.

Developments in wireless communication could change the CNAS landscape drastically. As with the hardware, tracking developments in communication technologies will remain important for future CNAS-like system development.

Scaling CNAS

In principle, CNAS is scalable to hundreds or even thousands of sensor agents deployed over a wide geographical area. In this contract we explored networks consisting of less than 20 agents (and simulated networks of up to 100 agents). Deploying large-scale CNAS networks will highlight issues in getting data and commands from one edge of the network across to the other edge. Should such communication become a problem, adding a secondary "long haul" repeater communication capability (one that is not IEEE 802.11-based) to a small subset of agents is a potential mechanism.

5 Conc1usion

CNAS has been successful as both a deployed agent-based sensor network and as a research environment. Working on CNAS has also been a lot of fun! Because CNAS is a "real" network, we could not ignore real-world engineering and conceptual challenges. Rather than hindering our research, these constraints forced us to reexamine all our intuitions and past practices and techniques. The result was a number of new findings and a large number of publications relative to the size of this contract. For example, sensor agents operating near the limit of radio-communication range with their radios turned off most of the time present different challenges than traditional MAS settings. This contradicted all work in "energy-efficient" communication that assumed that it was transmission that was energy expensive (based on message distance and volume), but that listening was free. Our emphasis during most of this effort has been on achieving basic CNAS capabilities and reliability, and we have reached a point where expanded use of the blackboard-system capabilities in CNAS and where incorporating more complex activities and adaptive reasoning into CNAS applications should be performed.

As this contract comes to a close, we have achieved an interesting dichotomy. On one hand, the PASTA microsensor-based CNAS network and ground-level atmospheric monitoring KSs have reached a level of maturity and stability that allows CNAS to be deployed quickly and confidently in the field. In in-the-field deployments, CNAS has demonstrated resiliency to hardware failures and even to periodic loss of all communication capability.

On the other hand, we have barely scratched the surface in terms of the cognitive capabilities that could be achieved with the blackboard-system architecture of each CNAS sensor agent. (We are using but a tiny percentage of the reasoning capabilities we have provided to each agent.) Our recent exploration of the feasibility of using new "commodity" processors (the PXA255-based Connex and the PXA270-based Verdex motherboards from Gumstix, Inc.) in CNAS, also suggests that cost-effective hardware for production deployments of CNAS-like networks is quickly becoming a reality. Coupled with our initial work on replenishable power and solar harvesting, these latest hardware directions will enable permanent deployment of CNAS agents in areas where the economic cost of losing a sensor agent due to damage or theft becomes tolerable. Our recent research in knowledgable routing and ideas for rapid radio cycling hold great promise for developing agent-based sensor networks that are both energy efficient *and* responsive. So, even with the success of the PASTA-based CNAS effort, significant opportunities for further research leading to substantial capability and performance improvements remain.

The future for agent-based sensor networks like CNAS is growing ever brighter.

Acknowledgments

CNAS would not exist without the efforts of Dr. Douglas Holzhauer (now retired from AFRL) who initiated the CNAS project and directed it through its formative months and PATRIOT 2006 deployment. Lok-Kwong Yan assumed direction of the UMass portion of the CNAS project following Doug's retirement.

Kevin Bartlett, Robert Fleishauer, Walter Koziarz, Wenchian Lee, Zenon Pryk, Wilmar Sifre, Lok-Kwong Yan, and Paul Yaworsky of AFRL/IF also contributed to the development and demonstrations of CNAS. Their patience, suggestions, and enthusiasm for this work were important to the progress we made in this effort. Kevin Bartlett, Douglas Holzhauer, Wenchian Lee, Wilmar Sifre, and Paul Yaworsky survived the summer heat and humidity at the PATRIOT 2006 Exercise. Walter Koziarz, Zenon Pryk, Lok-Kwong Yan, Wilmar Sifre, and Robert

Fleishauer supported the Talisman-Saber deployments of CNAS in Queensland, Australia.

AFRL loaned us several PASTA microsensors and two TACMET sensors for this work.

Carrie Kindler of ITT Industries developed the map-based 2D graphic display client used at the regional and console nodes.

Steven Hoffman of BBTech Corporation, working as an independent contractor on this effort, performed much of the tedious work of cross-compiling the CLISP `lisp.run` C-kernel for the Gumstix.

On the UMass Amherst side of things, graduate students Michael O'Neil and Huzaifa Zafar made contributions to CNAS. Michael performed early work on mobile sensor agents, developing a Common Lisp interface to control the Sony AIBO "dogs." Huzaifa performed the organization-based routing and the solar-harvesting model development research. Professors Victor Lesser and Deepak Ganesan advised Huzaifa on the organization-based routing research. Finally, MAS Laboratory Business Manager and Grant Administrator Michele Roberts kept the diverse administrative and reporting aspects of this collaborative effort under control and running smoothly.

Thanks to you all!

References

[1] Daniel D. Corkill, Douglas Holzhauer, and Walter Koziarz. Turn off your radios! Environmental monitoring using power-constrained sensor agents. In *Proceedings of the First International Workshop on Agent Technology for Sensor Networks* (ATSN-07), pages 31–38, Honolulu, Hawaii, May 2007.

[2] Daniel D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, September 1991.

[3] Robert S. Engelmore and Anthony Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

[4] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with >1000X dynamic power range. In *Information Processing in Sensor Networks* (ISPN 05), special track on Platform Tools and Design Methods for Network Embedded Sensors, Los Angeles, California, April 2005.

[5] Sandra E. Keene. *Object-Oriented Programming in Common Lisp*. Addison-Wesley, 1989.

[6] Gregor Kiczales, Jim des Rivieres, and Daniel G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[7] Daniel D. Corkill, Kevin Q. Gallagher, and Kelly E. Murray. GBB: A generic blackboard development system. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1008–1014, Philadelphia, Pennsylvania, August 1986. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 503–518, Addison-Wesley, 1988.).

[8] Daniel D. Corkill. Countdown to success: Dynamic objects, GBB, and RADARSAT-1. *Communications of the ACM*, 40(5):848–858, May 1997.

[9] Daniel D. Corkill. Reporting down under: A CNAS (Collaborative Network for Atmospheric Sensing) update. In *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks* (ATSN-08), Estoril, Portugal, May 2008. To appear.

[10] Daniel D. Corkill and Victor R. Lesser. The use of meta-level control for coordination in a distributed problem-solving network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, Karlsruhe, Federal Republic of Germany, August 1983. (Also published in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986.).

[11] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 2005.

[12] S. Kumar. Confidence based dual reinforcement Q-routing: An on-line adaptive network routing algorithm. Master's thesis, University of Texas at Austin, Austin, Texas, 1998.

[13] Huzaifa Zafar, Victor Lesser, Daniel Corkill, and Deepak Ganesan. Using organization knowledge to improve routing performance in wireless multi-agent networks. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS 2008), Estoril, Portugal, May 2008. To appear.

[14] Huzaifa Zafar and Daniel Corkill. Simplifying solar-harvesting model development in situated agents using pre-deployment learning and information sharing. In *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks* (ATSN-08), Estoril, Portugal, May 2008. To appear.

Appendices

A Installing and Running CNAS

The CNAS sensor-agent software is tightly coupled to the agent hardware and OS kernel. On the PASTA, an archive file containing the modified Debian CLISP, precompiled GBBopen, and CNAS-agent software is simply copied onto the PASTA (using `scp`) and extracted. On boot-up, the PASTA automatically compiles the latest CNAS code and starts the CNAS service.

Installing CNAS on a Console or Regional Node

Before installing the CNAS and GBBopen software, install a supported Common Lisp implementation.[25] The latest Subversion (SVN) snapshot archive for GBBopen can be found in the "Downloads" area of the GBBopen web site (`http://GBBopen.org/`). The on-line GBBopen hyperdoc reference and the GBBopen Reference Manual (in PDF format) can be found in the "Documentation" area of the web site. Links to supported Common Lisp implementations can be found on the "Current ports" page of the GBBopen web site. In either case, download or extract the GBBopen files to an installation directory of your choosing and follow the installation instructions in the GBBopen Tutorial that is included in the archive (and is also available on the GBBopen web site.)

Next, create a subdirectory named `gbbopen-modules` in your home directory. (Your "home" directory can be ambiguous on machines running a Microsoft Windows operating system. The result returned from the Common Lisp function `(user-homedir-pathname)` is the directory that your Common Lisp implementation considers to be your "home" directory.)

Next, extract the CNAS archive into a directory of your choosing.

Then, create a symbolic link in your `gbbopen-modules` directory (the directory that you created as a subdirectory of your "home" directory (above) that points to the directory where you installed CNAS (the directory where the `commands.lisp` and `modules.lisp` files for CNAS are found).

Start your Common Lisp and load the GBBopen's `initiate.lisp` file. (If you have set up GBBopen according to the "Enhancing your Development Environment" exercise in the *GBBopen Tutorial*, this will be loaded for you automatically when you start your Common Lisp.

Then, at the read-eval-print prompt, enter `(cnas-rn :create-dirs)` on the regional node. You only need to specify this the first time you install CNAS. The `:create-dirs` option tells GBBopen to automatically create directories for holding the compiled CNAS files. After this initial compilation, you can compile and load CNAS simply by entering the command `:cnas-rn` after the GBBopen's `initiate.lisp` file has been loaded.

Start the regional node by entering `(start)`. The regional node can be stopped by entering `:stop` at the command loop prompt.

On a console node, perform the same steps as above, except replace the form `(cnas-rn :create-dirs)` with `(cnas-cn :create-dirs)` and `:cnas-rn` with `:cnas-cn`.

[25] A CNAS console or regional node requires a multiprocessing Common Lisp. We used SBCL running on Linux in this effort

B Building CLISP on the Gumstix

Cross-compiling CLISP for the Gumstix involves a manual process of stepping through the CLISP build process, making some edits to build scripts and transferring pieces from the Linux/x86 cross-compilation host to the Gumstix target. As with all C-based software on the Gumstix, CLISP's `lisp.run` image is intimately tied to the Gumstix kernel and library software, so this procedure has to be performed with each change.

The following procedure was developed for buildroot. OpenEmbedded[26] (OE) came into favor after the end of the contract, so it has not been tested with OE (but it should work similarly, at least in theory).

Download the latest CLISP sources from `http://clisp.cons.org/` and extract to a directory that we will refer to as `clisp`. Change to this `clisp` directory and enter:

```
> make distclean
> ./configure --without-readline --host=arm-gumstix-linux clisp-arm
```

Then in `clisp/clisp-arm/` enter:

```
> ./makemake --with-dynamic-ffi --without-readline --srcdir=../src arm \
/home/gumstix/gumstix-buildroot/build_arm_nofpu/staging_dir/bin/arm-linux-gcc
```

Edit the created file `Makefile`. Comment out the existing `CFLAGS` and replace with:

```
CFLAGS = -O2 -DUNICODE -DDYNAMIC_FFI -DNO_READLINE -DNO_SIGSEGV -I. -DARM -UI80386
```

Save the file and then enter:

```
> make init
> make allc
> make lisp.run
```

This will create `lisp.run` which can be transferred onto the Gumstix and should execute there without error.

The next step is use CLISP on the Linux/x86 to create compiled lisp files for the ARM. The `.fas` files that serve to build CLISP's image are not portable; they contain platform-specific and build-specific code. However, the CLISP compiler can be used to generate cross-compiled `.fas` files by using the following form:

```
(let ((*features* (cons :arm (remove :unix *features*)))
      (system::*big-endian* nil))
  (compile-file "file1.lisp")
  (compile-file "file2.lisp")
    ...)
```

Each all of the files needed by CLISP can be compiled in this way and then transferred to the Gumstix. Note that this compilation only has to be done once for a specific CLISP release version; the same `*.fas` files can be used with different Connex and Verdex kernel and library verions.

Once compiled and transferred to the Gumstix, a memory image containing them needs to be created. Then it can be used as:

```
./lisp.run -M lispinit.mem
```

[26] http://www.openembedded.org

C Turn Off Your Radios! The ATSN-07 Paper

This appendix contains a copy of the ATSN-07 paper: "Turn Off Your Radios! Environmental monitoring using power-constrained sensor agents," by Daniel D. Corkill, Douglas Holzhauer, and Walter Koziarz. This paper appeared in the *Proceedings of the First International Workshop on Agent Technology for Sensor Networks* (ATSN-07), Honolulu, Hawaii, pages 31–38, May 2007.

Turn Off Your Radios!

Environmental Monitoring Using Power-Constrained Sensor Agents

Daniel D. Corkill
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01002
corkill@cs.umass.edu

Douglas Holzhauer* & Walter Koziarz
Air Force Research Laboratory
Rome, NY 13441-4505
douglas.holzhauer@sunyit.edu
Walter.Koziarz@rl.af.mil

ABSTRACT

CNAS[1] (Collaborative Network for Atmospheric Sensing) is an agent-based, power-aware sensor network for ground-level atmospheric monitoring. In many multi-agent applications, reducing message transmission is a primary objective. In CNAS, however, it's not the cost of sending messages, but *when* messages can be sent that is the driving communication constraint. CNAS agents must have their radios turned off most of the time, as even **listening** consumes significant power. Working in such *collaborative isolation* changes the character of agent interaction, as agents must have their radios turned on when others are sending messages to them. CNAS requires agent policies that can intelligently meet operational requirements while communicating only during intermittent, mutually established, communication windows.

In this paper, we describe the CNAS agents and their hardware and blackboard-system software architectures. We also relate experiences and lessons learned from a field deployment of CNAS at the 2006 PATRIOT Exercise held last July at Fort McCoy, Wisconsin, and we discuss the upcoming CNAS deployment in conjunction with the Talisman Saber Combined Exercise to be conducted May–July 2007 in Australia. We conclude with an overview of current CNAS research that is exploring the addition of a rollable solar panel to each sensor agent that allows its battery reserves to grow (up to full capacity) when sunlight is available. Replenishable power reserves can support unlimited operational lifetimes, but activity decisions become more complex as each agent now must consider how much additional power may become available and when.

1. ATMOSPHERIC MONITORING

The U.S. Air Force is interested in sensor networks for ground-level environmental monitoring, as detailed knowledge of local atmospheric conditions increases air drop precision and all-weather landing safety. Such networks also have application to detecting forest fires, monitoring their changing status, and informing firefighters of changing conditions that affect their strategy and safety. Similarly, detailed knowledge of local atmospheric conditions is important in managing responses to airborne hazardous materials (hazmat) incidents and in determining prudent evacuation areas and routes.

Low-level atmospheric phenomena are characteristically complex with changing spatial gradients. Because of this complexity, mathematical models based on a small number of observations can not accurately quantify important local environmental variations. At present, weather-based mission decisions use predictions made by the Air Force Weather Agency using complex large-scale models such as Mesoscale Model 5.1 (MM5).[2] Using the new Weather Research & Forecasting (WRF) model,[3] which incorporates individual observations into MM5, can increase the prediction accuracy. Currently most observations fed into MM5/WRF are acquired by satellites or by land based radar. Naturally, the closer the direct observations used as inputs to MM5/WRF are to the region of interest, the more accurate the predictions for that region will be. Even when used in combination, these large-area sensors can exhibit serious limitations when the area of interest is located in a remote isolated region. Cloud cover can mask the lower elevation weather parameters, and the curvature of the earth quickly restricts ground radars from observing lower portions of the troposphere. In the case of mountainous terrain, large geographical changes over small distances can prevent even the best models from accurately determining local weather conditions [3].

Large, battery-powered, ad hoc sensor networks can provide the high-accuracy environmental data needed in these application settings. Work by both DARPA and AFRL's Sensors Directorate is leading to sensor nodes that are sufficiently rugged that they can be air dropped into regions of interest[4] and that are able to selectively control their battery-power expenditures to provide monitoring services over extended time periods. Self-organizing, air-dropped sensor networks will enable the collection of detailed envi-

The UMass portion of this work is supported by the AFRL "Advanced Computing Architecture" program, under contract FA8750-05-1-0039. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views contained in this paper are the authors.'

*Doug Holzhauer is now retired and teaching at SUNYIT.

[1]Pronounced "see-nas."

[2]http://www.mmm.ucar.edu/mm5/

[3]http://www.wrf-model.org/

[4]The current tactical weather station used by the Air Force, the AN-TMQ-53, cannot be air dropped because of its cost and packaging.

ronmental data from regions that were previously closed to ground-level monitoring.

Air-dropping atmospheric monitoring nodes introduces additional issues. Normally when a weather station is positioned, meteorologists use their understanding of geography and meteorology to optimize the location for weather measurements. Precise placement is not possible, when sensor nodes are air dropped. (However, research and development into maneuverable air-drop delivery systems are underway.) For the time being, though, even a marginal location for an air-dropped weather station can not be assured. To compensate, additional sensors may be deployed and their observations weighted as to quality.

Environmental monitoring networks may also include many different types of sensors, and individual sensor capabilities may need to be dynamically adjusted (in terms of what aspects of the environment are sensed, the precision, power, and usage frequency of sensing, and the amount of local processing done by each sensor node before transmitting information). Information processing in the network may require the integration/fusing of information coming from heterogeneous and geographically distant sensors. Additionally, sensor usage and parameters may need to be adjusted in real-time as the network tracks phenomena moving through the environment and as the power and communication resources available to the sensor nodes change. Battery-powered sensor nodes need to spend their limited power wisely in collectively performing their best in achieving overall sensor-network goals.

In addition to this real-time, operational agility, the design of the sensor network should allow the software approaches and algorithms of nodes to be changed, improved, and extended throughout the operational lifetime of the network. We should expect from the outset that new and improved components and software techniques will be developed over time and added to the system. The underlying design of the sensor network should be able to adapt to such new capabilities and be able to manage their use effectively.

Sensor Agents

A central challenge in building effective environmental-monitoring sensor networks is coordinating the use of critical resources (including sensors, processing, communication, and power) to best achieve conflicting mission, organizational, and sensing goals [2]. In resource-constrained settings typified by sensor networks, the activity decisions of "who," "what," "when," "where," and "with whom" must involve an overall awareness of organizational and operational capabilities and goals, the state of activities and resources in the network, and the time-critical nature of activities and sensor data. This requires that every sensor node understand that it is part of a larger organization and that it may need to satisfy more global goals at the expense of its own local goals.

The need for autonomous and self-aware sensor nodes is a natural fit for employing multi-agent system (MAS) technology. Agent-based sensor networks are part of an important class of MAS applications in which issues of organizational structuring, coordination, collaboration, and distributed, real-time resource allocation are critical for success. Simply put: sensor agents must do more than react to their local situation—they must collaboratively determine what they should be doing, when they are doing them, and why.

2. CNAS

CNAS (Collaborative Network for Atmospheric Sensing) is an experimental, agent-based, power-aware sensor network for ground-level atmospheric monitoring. It is intended as a research and demonstration tool for conducting realistic explorations of the advantages and limitations of agent-based environmental monitoring using hardware capabilities that are likely to become cost-effective for production deployments in the next few years.

CNAS has the following major hardware and operational characteristics:

- Sensor agents have on-board GPS capabilities for obtaining location and time data.
- The distance separating sensor agents is near the limit of their wireless communication range. Alternate message-route options are relatively sparse, and much communication is multi-hop.
- The WiFi adapter used at each agent is the component with the largest power-expenditure rate, by far. So, although the power to the central processor and peripheral modules at each sensor agent can be controlled independently, such power savings are secondary to the savings achieved by having the WiFi turned off as much as possible.
- Sensor agents obtain and process local-environment readings once every second (even when their WiFi is powered off).
- Summaries of the sensor-agent readings taken over geographic region of interests are aggregated and maintained by sensor agents performing a regional "cluster head" service.
- Mobile "console nodes" may enter and leave the CNAS monitoring area, obtaining regional summaries and individual sensor-agent data and changing the tasking and policies of CNAS.
- The processing, memory, and storage capabilities available at each sensor agent are relatively powerful.

We use a combination of blackboard-system and MAS techniques in our CNAS sensor agents. Blackboard systems [4, 5] are proficient in supporting indirect and anonymous collaboration among software entities and in exploiting temporal decoupling of entity interactions in order to obtain maximum flexibility in coordinating activities. MAS researchers, on the other hand, have developed effective techniques for operating in highly distributed, dynamic settings and for coordinating local, autonomous activity decisions. These capabilities are all highly valuable assets in designing an effective architecture for agile, resource-aware sensor network agents. Nevertheless, applying blackboard and MAS approaches in concert to the sensor-net domain is a novel aspect of the CNAS effort.

The agent-level design of CNAS was driven by the hardware and support software that had been designated for this effort. Therefore, from a MAS perspective, the hardware and support-software characteristics and capabilities

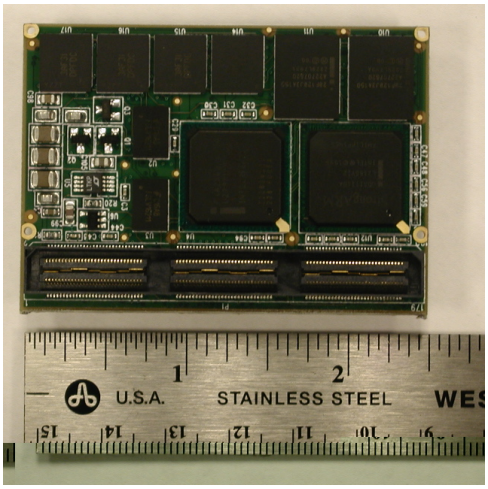


Figure 1: PASTA PXA255 CPU Module

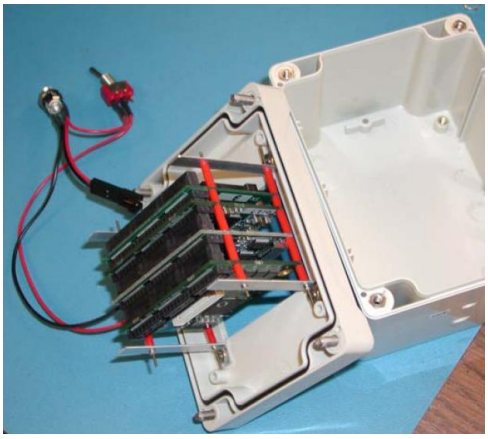


Figure 2: PASTA/Crossbow Sensor Agent

are pre-established and unchangeable. Our challenge was to develop an effective agent-based monitoring network using the specified hardware and operating-system software.

Sensor-agent hardware

Each sensor agent is built around ISI's PASTA (Power-Aware Sensing, Tracking, and Analysis) microsensor platform [8][5] and its Intel PXA255-based CPU (Figure 1). Unlike traditional hub-and-spoke sensor-node architectures that have peripherals clustered around a central processor, the PASTA uses a distributed-peer model that can decouple processing from peripheral operation. In the hub-and-spoke architecture, the central processor must be continually active to broker peripheral operations, and this power consumption represents the lowest possible rate of total system-power expenditure.

In the decoupled, distributed model used in the PASTA, the central processor and peripheral modules operate autonomously, and each can be powered independently. Higher-performance processing can be made available when needed, but low average-system-power expenditure can be

[5]http://pasta.east.isi.edu/

Figure 3: Crossbow MTS420CA

achieved by operating in extremely low-power modes with only essential modules active whenever possible. The PASTA's central processor is running a customized 2.4.19 Linux kernel, even though this places Linux in the unconventional role of a peer module rather than controlling a central processor.

In addition to the PASTA, each CNAS sensor node (Figure 2) is equipped with a Crossbow MTS420CA sensor board (Figure 3) providing:

- Intersema MS5534AM barometric pressure sensor
- TAOS TSL2550D ambient light sensor
- Sensirion SHT11 relative humidity/temperature sensor
- Leadtek GPS-9546 GPS module (SiRFstar IIe/LP chipset)
- Analog Devices ADXL202JE dual-axis accelerometer[6]

A Netgear MA111 Wireless adapter, connected to the PASTA's USB interface, provides standard IEEE 802.11b wireless communication, which achieves the 1–2km range required by CNAS.

Power expenditure & communication

The 12-volt battery used at each sensor agent provides approximately 12,000mA-hours of power. The IEEE 802.11b USB adapter is the component with the largest power-expenditure rate, by far, on the sensor agent. The adapter draws at a rate of 250mA when powered on, which would consume all battery power, operating alone, in 48 hours. Since we were constrained to use 802.11b in CNAS, the only solution was to have each agent turn off its WiFi adapter most of the time. These radio-power decisions cannot be made unilaterally, as other nodes need to know when their transmissions will be receivable, and nodes may be acting as forwarders in multi-hop transmissions that don't pertain to them.

Communication policies and routing protocols have been designed especially for energy saving in wireless sensor networks. (Akkaya and Younis provide a recent survey [1].) Most of these policies assume sensors are stationary, as is the case with CNAS. Some assume mobile sinks, like CNAS's console nodes, and periodic reporting requirements. Unlike CNAS, where listening is as expensive as sending and agents are at the limit of their direct communication range, much of the energy-efficient routing work focuses on limiting transmission quantity and distance while assuming full-time listeners. Even in protocols such as Geographic Adaptive Fidelity (GAF) [9], where nodes are switched on and off to reduce communication-energy expenditures, a percentage of the nodes in each geographic region are always on.

[6]Not used in CNAS.

In CNAS most, if not all, sensor agents must have their radios activated at the same time—if only to provide a multi-hop route for other agents. We address this in the obvious way, by using a set of compatible time-based radio-power policies that allow nodes that may not be aware of the current policy to eventually synchronize their policy with others. Each policy consists of fixed-length communication windows that occur at regular intervals, where the windows of each policy align with one another whenever possible. The policies we are using are:[7]

hourly: A communication window occurs at the top of each hour

half-hourly: A communication window occurs every 30 minutes, starting at the top of each hour

quarter-hourly: A communication window occurs every 15 minutes, starting at the top of each hour

hourly-overnight-sleep The hourly policy, but without communication after the 6PM window until the 6AM window the next morning (local time)

half-hourly-overnight-sleep The half-hourly policy, but without communication after the 6PM window until the 6AM window the next morning (local time)

quarter-hourly-overnight-sleep The quarter-hourly policy, but without communication after the 6PM window until the 6AM window the next morning (local time)

The current policy can be switched at the next communication window, based on current weather trends and mission objectives. A new or rebooted node that is not aware of the current policy can be assured of communicating with others during the next daytime top-of-the-hour window, no matter which policy is in effect. Alternatively, the node can be more aggressive and try connecting at the next quarter-hour window, and at subsequent fallback windows.

Inter-node communication in CNAS uses standard TCP/IP operating over an OLSR (Optimized Link State Routing)[8] multi-hop protocol. OLSR is intended for dynamic routing under changing connectivity and propagation conditions, and where a relatively small proportion of nodes are likely to come and go at the same time. Due to the long periods of no radio power, CNAS forces OLSR to essentially reinitialize at the start of each communication window. Given this stabilization requirement, each CNAS communication window was structured as the sequence of activities shown in Figure 4. These staged activity intervals are very conservative, and provide substantial slack time for OLSR re-initialization and for coping with highly degraded communication. They also can be changed on the fly, so shorter, more aggressive, communication windows can be attempted. During a communication window, each agent uses an application-level message retransmission strategy whenever the TCP/IP-layer reports delivery failure. A prioritized store-and-retry message delivery strategy holds outgoing messages that cannot be delivered due to outage during a communication window as well as messages generated when the agent's radio is turned off.

[7]An alternate set of policies was considered in which the half-hourly policies were replaced with windows occurring every 20 minutes and the quarter-hourly policies with windows occurring every 10 minutes. With significantly shorter communication windows, this alternate set, perhaps even augmented by an every-5-minute policy, might be preferable.

[8]http://olsr.org/

0–60 sec	WiFi power on, OLSR stabilization
60–120 sec	node assessment, status exchange, high-priority message delivery, cluster head determination
120–140 sec	node observation transmission (to cluster head)
140–240 sec	cluster head processing, low-priority message delivery
240–300 sec	cluster observation transmission (to console & regional nodes)
300 sec	WiFi shutdown

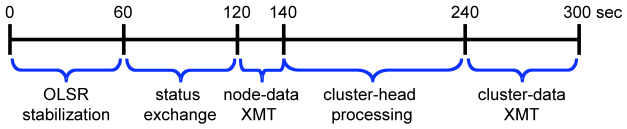


Figure 4: CNAS Communication Window

Software

One objective of the CNAS effort was demonstrating the feasibility of hosting a high-level language and an AI blackboard-system framework on the PASTA. After a preliminary assessment, we felt that it was indeed possible to support both Common Lisp and the GBBopen open-source blackboard-system framework[9] on the PASTA, and we began a porting effort to the PASTA for CNAS. GBBopen is written in Common Lisp and uses CLOS (the Common Lisp Object System) [6] and the Common Lisp Object System Metaobject Protocol (MOP) [7] to provide blackboard-specific object capabilities. The blending of GBBopen with Common Lisp transfers all the advantages of a rich, dynamic, reflective, and extensible programming language to blackboard-application developers. Thus, GBBopen's "programming language" includes all of Common Lisp in addition to the blackboard-system extensions provided by GBBopen.

We initially considered using a partially completed ARM port of SBCL (Steel Bank Common Lisp)[10] as the Common Lisp implementation for the PASTA. SBCL is an open-source Common Lisp implementation that supports an excellent optimizing native-code compiler. Although SBCL's compiler technology supports both RISC and Intel-class processors, the combination of a RISC instruction set with a relatively limited number of registers (more Intel like) on the ARM processor did not match any of the existing compilation models in the SBCL compiler. The need to implement a new "hybrid" compilation model for the ARM processor had delayed volunteer work on the finishing the ARM port of SBCL indefinitely.

We did not have the time or resources to invest in completing the ARM port, and were forced to abandon an SBCL strategy. Fortunately, at about this same time, sufficient MOP support for GBBopen was completed for another open-source Common Lisp implementation, CLISP.[11] CLISP was already ported to ARM processors, so with the added MOP support in CLISP 2.34, we had a viable Common Lisp implementation for hosting GBBopen on the PASTA.

[9]http://GBBopen.org/

[10]http://sbcl.sourceforge.net/

[11]http://clisp.cons.org/

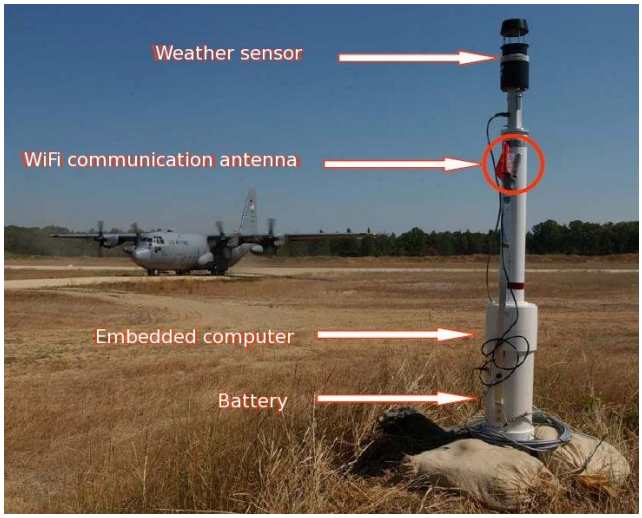


Figure 5: A TACMET-Augmented Sensor (at the PATRIOT 2006 exercise)

An important advantage of CLISP is that it is structured as a small C-based kernel that operates in conjunction with a platform-independent bytecode compiler and virtual-machine executer. A major disadvantage of CLISP, however, is that it does not support Lisp multiprocessing (process threads), which complicates real-time event processing. However, the small and portable C-based kernel and compact bytecode executer were well suited to the memory space available on the PASTA.

We were able to make use of the Debian ARM packaging of CLISP 2.34 (performed by Will Newton). The Debian package allowed us to bypass cross-compiling the CLISP kernel using an ARM cross-compilation toolchain running on an Intel x86 host and bootstrapping the rest of the build directly on the PASTA (which would have been a very painful process). Using the Debian package did introduce some problems, however. Differences between Debian and the PASTA's TinyOS Linux distribution required forced bypassing of package dependencies and introduced incompatibilities in several shared libraries. The latter resulted in some degradation in CLISP's memory management and garbage-collection performance on the PASTA.

The Pasta running basic Linux system processes and an `ssh` remote-login session has approximately 34.3MB of free memory space (out of a total of 64MB). CLISP consumes slightly more than 2MB, and GBBopen uses another 2MB. This brings the free memory down to about 30MB that is available for blackboard objects, knowledge sources (KSs), and sensor data—a reasonable amount for performing sensor-agent processing.

Node types & roles

A CNAS network can contain four different "types" of agent-based nodes:

Sensor Agents: A basic CNAS sensor agent consists of: the PASTA stack and Crossbow, a USB WiFi adapter, and a 12V battery, all packaged in a PVC housing that positions the wireless antenna and sensors 4.25' above ground level. The node packaging is intentionally large to allow easy access during testing and evaluation.

TACMET-Augmented Sensor Agents: A TACMET-augmented sensor agent is a basic CNAS sensor agent (as above) that also includes a Climatronics TACMET II 102254 weather sensor (see Figure 5). The TACMET II provides a temperature sensor, a fast-response, capacitive relative humidity sensor, a barometric pressure sensor, a flux gate compass, and a folded-path, low-power sonic anemometer. Wind speed, wind direction (resolved to magnetic North with the flux gate compass), temperature, and relative humidity readings are provided to the PASTA over an RS-232C serial connection once every second. Power to the TACMET cannot be controlled by software, and the TACMET is powered by a separate (unmanaged) 12V battery. TACMET II provides wind speed and direction readings that are not available from the Crossbow sensors, as well as corroborating measurements for temperature, humidity, and pressure. Unlike the Crossbow, TACMET II measurements have been certified by the Air Force, and having duplicate measurements allows calibration of the Crossbow readings.

Console Nodes: A console node is a laptop or handheld computer that, upon entry into the CNAS-network area, can obtain observation data from the network and display that data graphically. Unlike sensor and TACMET nodes, console nodes do not turn their wireless on and off. An authorized user at a console node can change network objectives and policies, transition the network into continuous-communication ("debugging") mode, and perform detailed inspection of the data and activities at individual sensor agents. Of course, unless the network is in continuous-communication mode, these activities can only be performed during communication windows when sensor agents have their radios turned on.

Regional Node: The regional node is a console node that is also connected to an external network, such as the Internet. When a regional node is available in the CNAS network, observational data and summaries can be made available outside the CNAS monitoring region, and authorized remote users can re-task CNAS objectives, perform detailed inspection of the data and activities at an individual sensor agent, and even update sensor-agent software.

Each CNAS sensor agent performs basic sensing activities, obtaining atmospheric readings once each second. Following Air Force meteorological practice, the following summary readings are computed from the 1-second readings and saved every five minutes:

- temperature (5-minute average)
- dew point (5-minute average)
- pressure (last reading)
- altimeter (last reading)
- wind-u-component (2-minute average, TACMET agents)
- wind-v-component (2-minute average, TACMET agents)

All unsent 5-minute summary observations are transmitted to the node acting as the *cluster head* (discussed shortly) for the agent during the next communication window.

In addition to the 5-minute summary observations, each sensor node performs an interval-based compression of the raw sensor observations. These compressed 1-second readings and the 5-minute summary observations are held by the agent for a user-specified period (typically for many days).

Each sensor agent performs saturation-vapor-pressure, humidity-to-dew-point, pressure-to-altimeter, millibars-to-inches, and wind-meter-to-knot computations as needed. The PASTA does not include floating-point hardware, so these computations are performed by software emulation.

Cluster heads

In addition to its sensor duties, a regular sensor or TACMET agent can also assume the role of cluster head. A *cluster* in CNAS is a grouping of sensor nodes located in a geographic region of interest. Non-overlapping cluster regions are user-defined and are communicated to all nodes from a console or regional node. Each node determines its cluster membership at start up, based upon its location and the user-specified cluster regions it receives when it first makes contact with another node in the network. Through an information-spreading process, the identity and locations of all nodes in a node's cluster becomes known. Given this cluster-membership information and globally established criteria, each agent computes a total preference ordering over all the agents in its cluster for assuming the cluster-head role.

During the initial phase of every communication window, each sensor agent determines the agent that is the most preferred cluster-head among all the agents that are alive and communicating in its cluster. If the agent is not assuming the cluster-head role, it transmits its 5-minute observations to the cluster head. If it is the cluster head, it accumulates the observations received from the other agents in its cluster, creating 5-minute cluster summary observations.[12] As the end of the communication window draws near (at the 4-minute mark in our conservative policy), the cluster head transmits the cluster summaries to all console and regional nodes that are active.

The cluster-head determination policy takes advantage of the OLSR-layer routing information to determine what nodes can receive messages from the agent. Should a cluster become bifurcated, separate cluster heads will be selected for each cluster fragment. When connectivity is re-established, these cluster heads provide summaries for the same cluster to console and regional nodes, where they can be combined into a single cluster summary. Furthermore, the most preferred sensor agent will again become the sole head of the reunited cluster.

3. CNAS DEPLOYMENT: PATRIOT 2006

CNAS was field tested at the 2006 PATRIOT Exercise held last July at Fort McCoy, Wisconsin. Over 1,600 Army and Air National Guardsmen, U.S. Air Force and Army active-duty and Reserve personnel, and soldiers and airmen from Canada, the Netherlands and the United Kingdom participated in the Exercise. Nine sensor agents and 8 TACMET-augmented sensor agents were manually positioned (not air dropped) in the area around Young Field and the Badger Drop Zone (see map, Figure 6). A telephone line at the south-eastern edge of the monitoring area was also reserved to connect a laptop-based regional node to the Internet via a dial-up modem connection.[13]

[12]Cluster summaries include the list of the individual nodes that contributed to them.

[13]The regional node was removed each night.

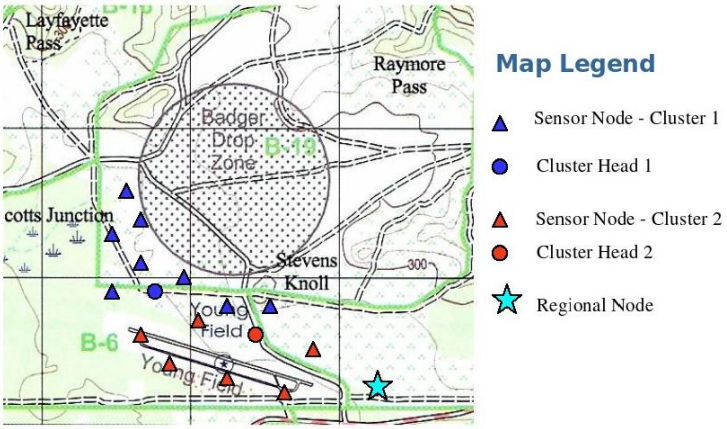


Figure 6: PATRIOT Sensor Agent Locations

Figure 7: Node 3 at the PATRIOT Exercise

Heat, humidity, line noise, and bugs

Our original plan was to demonstrate full CNAS capability at the PATRIOT Exercise. As the date of the exercise approached, firmware issues involving the Crossbow interface made its reliable operation uncertain, and we decided to deploy CNAS with the Crossbow sensors disabled. Without the Crossbow, GPS positioning and system-clock setting was lost. To compensate, a handheld GPS unit was used to determine the location of each sensor agent as it was placed, and this location and the node name (IP address) was entered into a console-node laptop. Then, as each sensor agent came on line, it obtained its location from the console node.

The PASTA does not have a hardware clock; the system clock has to be set every time the PASTA is booted. Without a GPS-obtained time, our fallback was to use the regional node as a CNAS time server and synchronize all agent clocks to it. This meant that when a sensor agent is booted, it does not have the correct time for synchronizing with CNAS network communication windows. We could

have implemented a strategy of cycling the rebooted node's radio on and off every few minutes until a communication window was observed, but we elected to have the node keep its radio active until it detected the presence of another node and then obtain the regional-node-based time from it.

The PATRIOT deployment began on the wrong foot, as it soon became clear that the provisioned Internet connection for the regional node was unusable due to high noise levels on the telephone line. This meant that one of our objectives, providing cluster-level METAR reports to external weather centers, would not be possible. We also had intended to allow authorized remote users to perform the same console-node CNAS commands as would be available if they were present in the monitoring area. Another objective lost to land-line quality.

We also had initial problems using the regional node as a network time server, which made synchronized communication windows difficult to achieve. Even using `ntpdate` with a 30-second timeout and a single sample, we had problems obtaining the time from the regional node. After some frustration, we discovered that the OLSR parameters at a number of the sensor agents had been set incorrectly, and that very few nodes were communicating beyond direct hops.[14] Once the parameter settings were corrected, the PASTA clocks became synchronized (at least within a few seconds of one another), and CNAS communication and cluster-head selection began to operate as intended.

With their Crossbow sensors deactivated, only TACMET agents could sense the environment. However, the other agents could still serve as cluster heads, and they still contributed to network connectivity (and therefore, still needed to participate in communication window activities). Originally only four TACMET II agents were planned, but without the Crossbow sensing, an additional four TACMET sensors were procured for the Exercise. Each sensor agent automatically detects if it has an operational TACMET sensor attached and, if it does, the agent assumes the TACMET-augmented role. However, there was another surprise in store for us. All four newly arrived TACMET sensors were producing garbled output. We initially feared that the new sensors had been damaged in transit, but we soon discovered that the serial output format of the new TACMET sensors was different from the original TACMETs—even though they were the same TACMET II model and part number as the originals.[15]

Fortunately, we had developed a live-updating facility for CNAS sensor agents. This facility allowed new or updated software to be distributed from a regional or console node to all CNAS agents during the next communication window. Taking advantage of the dynamic nature of Common Lisp, these updates are compiled and integrated directly in each running agent. The original plan was to test any such updates on a two sensor-agent CNAS network located in Amherst, Massachusetts. Tested updates would then be transmitted to the regional node at the PATRIOT Exercise via the telephone-line connection. However, with the land line unusable, we had to resort to transporting the regional node to the hotel (a one-hour trip), downloading the tested updates onto the regional node, and then returning the regional node to the Exercise site (another one-hour trip). As an additional complication, there were no TACMETs on the mini-network in Amherst—they were all at the Exercise! Nevertheless, co-authors Doug Holzhauer and Walt Koziarz used remote debugging mechanisms that had been put in place in CNAS to obtain the detailed serial device output from one of the remote sensor agents that was equipped with a new TACMET sensor. This information was later relayed verbally from the hotel by phone to Amherst. Once the updates supporting the new TACMETs were developed, transferred to the regional node, and then distributed to all CNAS nodes, all TACMET-augmented agents were sensing their surroundings.

During the exercise, transient hardware failures occurred in nearly one-half (6 of 17) of the sensor nodes. These failures occurred over several days when unseasonable air temperatures reached the upper 90s, and high humidity levels produced heat indexes approaching 110°F. These six sensor nodes returned to full functionality when the temperature dropped. Two other nodes failed permanently during the PATRIOT deployment. These failure rates were not unexpected, as many of the components used in the CNAS sensor nodes and the construction methods employed and were not intended to be used in such harsh surroundings.

In addition to hardware failures and software bugs, the PATRIOT deployment involved coping with *real* bugs. The worst of these were swarms of "ravenous" grasshoppers inhabiting the Fort McCoy area. They ate all of the surveyor flags that had been affixed to the sensor-node enclosures for visibility improvement. The grasshoppers also enjoyed wire insulation, and some even took up residence within the PASTA computer box, fancying the space between the boards comprising the processor and modules stack.

Even with these many issues, the PATRIOT deployment was a success. Sensor agents adapted to communication and node failures, reassigned cluster-head roles as needed, and provided local atmospheric data as designed. One afternoon, a regional tornado watch forced the cancellation of all activities and the withdrawal of troops and personnel (and the regional node!) from the area. CNAS sensor agents remained on duty, and when the regional node was activated the next morning, atmospheric data of the strong front's passage was provided.

4. NEXT DEPLOYMENT: AUSTRALIA

Based on the PATRIOT performance, CNAS has been selected for demonstration at the Talisman Saber Combined Exercise to be conducted May–July 2007 in Australia. Talisman Saber is a biennial series of joint exercises aimed at further developing and enhancing the defense relationships between the United States and Australia. With over 15,000 U.S. and 12,000 Australian personnel scheduled to participate, it is the largest and highest priority Tier II exercise in

[14]This highlights the difficulty of fully testing a sensor network like CNAS prior to deployment. The characteristics of widely scattered nodes cannot be duplicated accurately—even with sensor nodes distributed around research laboratory buildings.

[15]We learned later that the new TACMETs include a battery status report in their output.

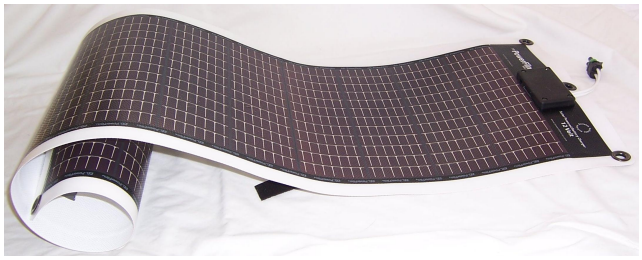


Figure 8: Rollable Solar Panel

the Pacific Theater.

One change from the PATRIOT deployment will be the full use of Crossbow sensing. We will use the MTS420CA's GPS unit to obtain the node location and set the clock. The GPS unit will be turned off most of the time, and only reactivated occasionally to correct clock drift and check that no one has moved the sensor node. We also plan to experiment with shorter communication windows than we used at PATRIOT. Since it will be winter, heat failures should not be a problem. (We do wonder what unexpected animal or insect life will complicate matters "down under.")

5. REPLENISHABLE POWER RESERVES

Sensor agents whose power reserves can be expected to be replenished from time to time adds new challenges in power management. We are currently exploring the addition of a rollable (thin film on plastic) solar panel (see Figure 8) to each sensor agent that allows its battery reserves to grow (up to full battery capacity) if the agent is in an unshaded location and the sun is shining. The activity decisions that are made by agents with replenishable resources now must consider how much additional power may become available and when. We assume that the CNAS sensor network is provided with the sunshine forecast (from sources outside the network), but each agent needs to learn the implications of the forecast on its own power reserves. A sensor agent that recognizes that it is shaded by a hill in the afternoon should realize that it cannot expect to obtain much power replacement if the forecast is for cloudy weather until noon. On the other hand, an unshaded agent could anticipate being able to perform additional power-intensive activities, based on its expectation of power replenishment.

6. SUMMARY

Developing CNAS has been exciting and challenging. Sensor agents operating near the limit of radio-communication range with their radios turned off most of the time lowers power expenditure at the cost of network responsiveness. Although CNAS can shift its communication policies in response to observations or planned objectives, it can be frustrating to have to wait until the next communication window in order to obtain CNAS data or transition all agents into continuous-communication mode. As developers, however, such frustrations are somewhat mitigated by the level of software capability that we have been able to achieve at each agent. PASTA processors running Linux, Common Lisp, and GBBopen provide a level of programming expressivity and on-the-fly modification that greatly facilitates the implementation of advanced behaviors and opportunistic-control decision making. We are fortunate in CNAS to have sufficient processing power and memory available to make shoe-horning complex behaviors and reasoning into tightly constrained C (and assembly) code a distant memory. Our emphasis to date has been on demonstrating basic CNAS capabilities and reliability, and we are eager to begin incorporating more complex activities and adaptive reasoning into CNAS in the coming months. We are also working to shorten the stabilization portion of the communication window by providing initial routing estimates at WiFi on time that are based on greater information sharing between the network routing and application layers.

The computational and power-management capabilities of the next generation of microsensor platforms will be even greater.[16] Perhaps someday soon, in a new generation of CNAS-like networks with advanced, low-power, long-distance communication hardware, we'll be able to say: "Leave your radios on!"

Acknowledgments

Kevin Bartlett, Wilmar Sifre, Wenchian Lee, and Paul Yaworsky of AFRL also contributed to CNAS. ISI's Joe Czarnaski and Brian Schott provided PASTA microsensor advice and assistance.

7. REFERENCES

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3(3):325–349, 2005.

[2] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug. 2003.

[3] D. A. Clark and M. P. Matthews. An integrated sensor testbed for support of theater operations in areas of complex terrain. In *Proceedings of the Battlespace Atmospheric and Cloud Impacts on Military Operations Conference (BACIMO 2000)*, pages 25–27, Fort Collins, Colorado, Apr. 2000.

[4] D. D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, Sept. 1991.

[5] R. S. Engelmore and A. Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

[6] S. E. Keene. *Object-Oriented Programming in Common Lisp*. Addison-Wesley, 1989.

[7] G. Kiczales, J. des Rivieres, and D. G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[8] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with >1000X dynamic power range. In *Information Processing in Sensor Networks (ISPN 05), special track on Platform Tools and Design Methods for Network Embedded Sensors*, Los Angeles, California, Apr. 2005.

[9] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 70–84, Rome, Italy, July 2001.

[16] PXA270-based modules are currently in Alpha testing.

D Reporting Down Under: The ATSN-08 Paper

This appendix contains a preprint of the ATSN-08 paper: "Reporting Down Under: A CNAS (Collaborative Network for Atmospheric Sensing) Update," by Daniel D. Corkill. This paper will appear in the *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks* (ATSN-08), Estoril, Portugal in May 2008.

Reporting Down Under

A CNAS (Collaborative Network for Atmospheric Sensing) Update

Daniel D. Corkill
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01002
corkill@cs.umass.edu

ABSTRACT

CNAS (Collaborative Network for Atmospheric Sensing) is an agent-based, power-aware sensor network for ground-level atmospheric monitoring. To conserve battery power, CNAS sensor agents must have their WiFi radios turned off most of the time, as even *listening* consumes significant power. This complicates agent interaction and system responsiveness, because an agent cannot simply turn on its radio when it needs to send a message. CNAS agents also must have their radios turned on when others are sending messages to them and to support multi-hop message forwarding.

In this paper, we briefly review the sensor-agent hardware and blackboard-system software used in CNAS, as well as how CNAS agents collaborate with only periodic radio availability. We also relate experiences and lessons learned from field deployments of CNAS at the Talisman-Saber Combined Exercise held in Queensland, Australia. We conclude with an overview of CNAS research performed since Talisman-Saber that focuses on: 1) improving CNAS performance and responsiveness with limited radio availability, 2) power-aware reasoning associated with solar harvesting obtained from a rollable solar panel at each sensor agent, and 3) potential next-generation CNAS hardware.

1. CNAS

CNAS[1] (Collaborative Network for Atmospheric Sensing) is an experimental, agent-based, power-aware sensor network for ground-level atmospheric monitoring [3]. CNAS is a research and demonstration tool developed jointly by AFRL (Rome, NY) and UMass for conducting realistic explorations of the advantages and limitations of agent-based environmental monitoring. The CNAS effort is investigating the use of hardware capabilities that are likely to become cost-effective for production deployments in the next few years.

A combination of blackboard and multi-agent system (MAS) techniques are used in CNAS sensor agents. Blackboard systems [2, 5] are proficient in supporting indirect and anonymous collaboration among software entities and in exploiting temporal decoupling of entity interactions in order to obtain maximum flexibility in coordinating activities. MAS researchers, on the other hand, have developed effective techniques for operating in highly distributed, dynamic settings and for coordinating local, autonomous activity decisions. These capabilities are all highly valuable assets in developing an effective software architecture for agile, resource-aware sensor network agents.

Each CNAS sensor agent is running Linux, Common Lisp, and the GBBopen blackboard-system framework.[2] Developing CNAS agent software using Common Lisp and GBBopen provides a level of programming expressiveness and on-the-fly modification that greatly facilitates the implementation of advanced behaviors and opportunistic-control decision making.

Agent types

A CNAS network contains four different "types" of agents:

Sensor Agents: Each sensor agent is built around ISI's PASTA (Power-Aware Sensing, Tracking, and Analysis) microsensor platform [8][3] and its Intel PXA255-based CPU. In addition to the PASTA, each sensor agent is equipped with a Crossbow MTS420CA sensor board containing:

- Intersema MS5534AM barometric pressure sensor
- TAOS TSL2550D ambient light sensor
- Sensirion SHT11 relative humidity/temperature sensor
- Leadtek GPS-9546 GPS module (SiRFstar IIe/LP chipset)
- Analog Devices ADXL202JE dual-axis accelerometer[4]

A Netgear MA111 Wireless adapter, connected to the PASTA's USB interface, provides standard IEEE 802.11b wireless communication. This hardware, and a 12V battery, is packaged in a PVC housing that positions the wireless antenna and sensors 4.25' above ground level. The agent packaging is intentionally large to allow easy access during testing and evaluation.

Each CNAS sensor agent performs basic sensing activities, obtaining atmospheric readings once each second. Following Air Force meteorological practice, the following summary readings are computed from the 1-second readings and saved every five minutes:

- temperature (5-minute average)
- dew point (5-minute average)

The UMass portion of this work is supported by the AFRL "Advanced Computing Architecture" program, under contract FA8750-05-1-0039. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views contained in this paper are the authors.'

[1]Pronounced "see-nas." The original CNAS acronym was short for Cognitive Network for Atmospheric Sensing, but "Cognitive" has since evolved into "Collaborative," emphasizing the importance of the agent-based interactions supporting the *cognitive* reasoning activities in CNAS.

[2]http://GBBopen.org/

[3]http://pasta.east.isi.edu/

[4]Not used in CNAS.

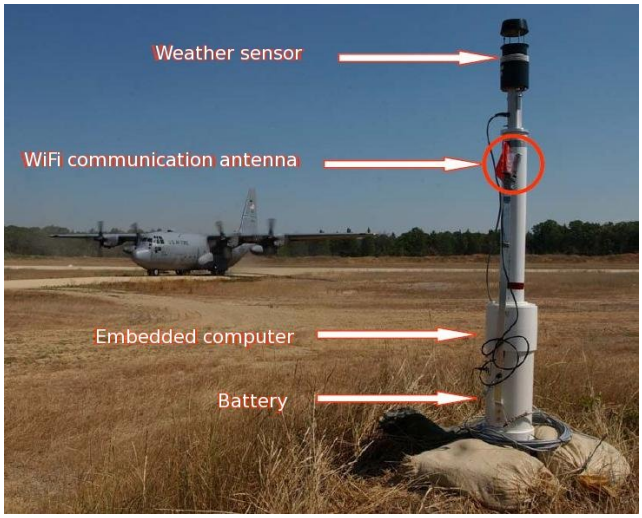


Figure 1: A TACMET-Augmented Sensor (at the PATRIOT 2006 exercise)

- pressure (last reading)
- altimeter (last reading)
- wind-u-component (2-minute average, TACMET agents)
- wind-v-component (2-minute average, TACMET agents)

Sensor agents perform saturation-vapor-pressure, humidity-to-dew-point, pressure-to-altimeter, millibars-to-inches, and wind-meter-to-knot computations as needed.[5] All unsent 5-minute summary observations are transmitted during the next communication window to the sensor agent that is acting as the *cluster head* (discussed shortly).

In addition to the 5-minute summary observations, each sensor node performs an interval-based compression of the raw sensor observations. These compressed 1-second readings and the 5-minute summary observations are held by the agent for a user-specified period (typically for many days).

TACMET-Augmented Sensor Agents: A TACMET-augmented sensor agent is a basic CNAS sensor agent (as above) that also includes a Climatronics TACMET II 102254 weather sensor (see Figure 1). The TACMET II provides a temperature sensor, a fast-response, capacitive relative humidity sensor, a barometric pressure sensor, a flux gate compass, and a folded-path, low-power sonic anemometer. Wind speed, wind direction (resolved to magnetic North using the flux gate compass), temperature, and relative humidity readings are provided to the PASTA over an RS-232C serial connection once every second. Unlike the Crossbow, TACMET II measurements have been certified by the Air Force.

Console Nodes: A console node is a laptop or handheld computer that, upon entry into the CNAS-network area, can obtain observation data from the network and display that data graphically. Unlike basic and TACMET-augmented sensor agents, console nodes do not turn their wireless on and off. An authorized user at a console node can change network objectives and policies, transition the network into continuous-communication ("debugging") mode, and perform detailed inspection of the data and activities at individual sensor agents. Of course, unless the network is in continuous-communication mode, these activities can only be performed during communication windows when sensor agents have their radios turned on.

Regional Node: The regional node is a special console node that is also connected to an external network, such as the Internet. When a regional node is available in the CNAS network, observational data and summaries can be made available outside the CNAS monitoring region, and authorized remote users can re-task CNAS objectives, perform detailed inspection of the data and activities at an individual sensor agent, and even update the CNAS software at individual sensor agents.

[5]The PASTA does not include floating-point hardware, so these computations are performed by software emulation.

Communication policies

Communication policies and routing protocols have been designed especially for energy saving in wireless sensor networks. (Akkaya and Younis provide a recent survey [1].) Most of these policies assume sensors are stationary, as is the case with CNAS. Some assume mobile sinks, like CNAS's console nodes, and periodic reporting requirements. Unlike CNAS, where listening is as expensive as sending and agents are at the limit of their direct communication range, much of the energy-efficient routing work focuses on limiting transmission quantity and distance while assuming full-time listeners. Even in protocols such as Geographic Adaptive Fidelity (GAF) [9], where nodes are switched on and off to reduce communication-energy expenditures, a percentage of the nodes in each geographic region are on at any point in time.

In CNAS most, if not all, sensor agents must have their radios activated at the same time—if only to provide a multi-hop route for other agents. We address this in the obvious way, by using a set of compatible time-based radio-power policies that allow agents that may not be aware of the current policy to eventually synchronize their policy with others. Each policy consists of fixed-length communication windows that occur at regular intervals, where the windows of each policy align with one another whenever possible. The policies used in the CNAS deployments are:

hourly: A communication window occurs at the top of each hour

half-hourly: A communication window occurs every 30 minutes, starting at the top of each hour

quarter-hourly: A communication window occurs every 15 minutes, starting at the top of each hour

hourly-overnight-sleep The hourly policy, but without communication after 6PM until 6AM (local time)

half-hourly-overnight-sleep The half-hourly policy, but without communication after 6PM until 6AM (local time)

quarter-hourly-overnight-sleep The quarter-hourly policy, but communication after 6PM until 6AM (local time)

The current policy can be switched at the next communication window, based on current weather trends and mission objectives. A new or rebooted agent that is not aware of the current policy can be assured of communicating with others during the next daytime top-of-the-hour window, no matter which policy is in effect. Alternatively, the agent can be more aggressive and try connecting at the next quarter-hour window, and at subsequent fallback windows.

Inter-agent communication in CNAS uses standard TCP/IP operating over an OLSR (Optimized Link State

0–60 sec	WiFi power on, OLSR stabilization
60–120 sec	agent assessment, status exchange, high-priority message delivery, cluster head determination
120–140 sec	observation transmission (to cluster head)
140–240 sec	cluster head processing, low-priority message delivery
240–300 sec	cluster observation transmission (to console & regional nodes)
300 sec	WiFi shutdown

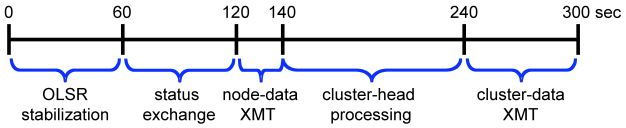


Figure 2: CNAS Communication Window

Routing)[6] multi-hop protocol. (Use of OLSR was an externally imposed requirement for the CNAS effort.) OLSR is intended for dynamic routing under changing connectivity and propagation conditions, and where a relatively small proportion of nodes are likely to come and go at the same time. Because the radio has been powered off, OLSR reinitializes at the start of each communication window. Given this initialization requirement, each CNAS communication window was structured as the sequence of activities shown in Figure 2. The durations of these staged activity intervals are very conservative and provide substantial slack time for OLSR re-initialization and for coping with highly degraded communication. During a communication window, each agent uses an application-level message retransmission strategy whenever the TCP/IP-layer reports delivery failure. A prioritized store-and-retry message delivery strategy holds outgoing messages that cannot be delivered due to outage during a communication window as well as messages generated locally when the agent's radio is turned off.

Cluster heads

In addition to its sensor duties, a regular or TACMET-augmented sensor agent can also assume the role of cluster head. A *cluster* in CNAS is a grouping of sensor agents located in a geographic region of interest. Non-overlapping cluster regions are user-defined, and each agent determines its cluster membership at start up, based upon its location and the specified cluster regions. Through an information-spreading process, the identity and locations of all agents in a cluster become mutually known. Given this cluster-membership information and globally established criteria, each agent computes a total preference ordering over all the agents in its cluster for assuming the cluster-head role.

During the initial phase of every communication window, each sensor agent determines the agent that is the most preferred cluster-head among all the agents that appear to be alive in its cluster. If a sensor agent is not assuming the cluster-head role, it transmits its 5-minute observations to the cluster head. If it is the cluster head, it accumulates the observations received from the other agents in its cluster, creating 5-minute cluster summary observations.[7] As the end of the communication window draws near (at the 4-minute mark in our conservative policy), the cluster head transmits the cluster summaries to all console and regional nodes that are active.

The cluster-head decision strategy takes advantage of the OLSR-layer routing information to determine what agents can receive messages from the agent. Should a cluster become bifurcated, separate cluster heads will be selected for each cluster fragment. When connectivity is re-established, these cluster heads provide summaries for the same cluster to console and regional nodes, where they can be combined into a single cluster summary. Furthermore, the most preferred sensor agent will again become the sole head of the reunited cluster.

Figure 3: Assembling a Sensor Agent

2. TALISMAN-SABER DEPLOYMENTS

In July 2006, CNAS was field tested at the 2006 PATRIOT Exercise held at Fort McCoy, Wisconsin. Based on the PATRIOT performance [3], CNAS was invited to participate at the 2007 Talisman-Saber Combined Exercise in Queensland, Australia. Talisman-Saber is a biennial Australia/United States bilateral exercise and the primary training venue for Commander Seventh Fleet Combined Task Force (CTF) operations. Over 15,000 Australian and US forces participated in Talisman-Saber 2007.

At the 2006 PATRIOT Exercise, problems with the PASTA firmware interface to the Crossbow forced us to deploy CNAS at PATRIOT with the Crossbow sensors disabled. An important goal for the Talisman-Saber deployment was to have the Crossbow fully operational. Unfortunately, ISI was unable to resolve the PASTA firmware issue (see Section 5), and we were forced, once again, to operate without Crossbow sensors.

Without the Crossbow, GPS positioning and system-clock setting[8] is also lost. To compensate, a handheld GPS unit was used to determine the location of each sensor agent as it was placed, and this location and the agent name were entered into a console-node laptop and then transferred to a configuration file on the sensor agent. In addition, since agents could not obtain the time using GPS, we used the re-

[6]http://olsr.org/

[7]Cluster summaries include the list of the individual agents that contributed to them.

[8]The PASTA does not have a hardware clock.

Figure 4: TACMET-Augmented Sensor Agent at the Talisman-Saber Drop Zone (Note the "high-tech" broom-handle antenna mast to the left of the sensor.)

Figure 5: CNAS Sensor Agents in Transport

gional node as a CNAS time server. This meant that when a sensor agent is booted, it does not have the correct time for synchronizing with CNAS network communication windows. We could have implemented a strategy of cycling the rebooted agent's radio on and off every few minutes until a communication window was observed, but we elected to have the agent keep its radio active until it detected the presence of another agent and then obtain the regional-node-based time from it.

Drop-zone deployment

The CNAS network was initially deployed at Talisman-Saber at a remote[9] aerial drop zone (Figure 3). Conditions at the drop zone were austere: no electrical power, no telephone communication, and the shelter for the regional node and staff consisted of a floor-less tent erected in a field of brush adjacent to the drop zone. The CNAS objective was to gather and report a 72-hour window of historic low-level atmospheric data to aid in airdrop operations scheduled for June 19, 2007. Unlike the PATRIOT 2006 deployment which had two clusters, only a single cluster was defined in each of the Talisman-Saber deployments.

CNAS sensor agents were deployed in a linear line at the drop zone, with TACMET-augmented sensor agents interleaved with basic sensor agents (with disabled Crossbows). The distance between agents was almost 300m (at the limit for 802.11b). Position decisions were made in the field by walking with a laptop from the last positioned agent until its WiFi signal was lost, then walking backward a few meters and placing the sensor agent.

One technical issue that arose was the taller than anticipated native vegetation interfering with the 802.11b wireless communication used by CNAS agents. This issue was resolved by moving the WiFi adapters from the existing antenna masts to broom handles (obtained from a not so nearby hardware store) which raised the adapters high enough to diminish the native-vegetation attenuation (Figure 4).

Unlike the previous CNAS deployment at the 2006 PATRIOT Exercise—where the hardware in two sensor agents failed permanently and 6 of the remaining 17 sensor agents experienced transient hardware failures due to the unseasonably high air temperatures and humidity levels during the deployment—all CNAS agents operated flawlessly throughout all Talisman-Saber deployments. We attribute this to the elimination of faulty PASTAs during the 2006 PATRIOT "burn in" (literally) experience, improved agent-construction methods, and the more moderate *winter* weather in July in Queensland. The swarms of grasshoppers that took up residence within the PASTA computer boxes at PATRIOT 2006 (fancying the space between the processor and module boards) and that ate wire insulation within sensor agents were also missing from the Queensland deployments. Thankfully, no native creatures caused problems at Talisman-Saber.

Problems did arise with the regional-node laptop, however. The sound chip in the laptop failed and began spitting out sporadic interrupts, which interfered with the timing in the laptop's internal WiFi adapter. Because the bursts of spurious interrupts were unpredictable, connections might work for a while and then suddenly die (after broadcasting a number of packets with incorrect time values, confusing the OLSR collision-avoidance scheme at other agents). Once diagnosed, a backup laptop was used as the regional node. However, the backup laptop did not have a car charger, so without line power at the drop zone, the replacement regional node had to operate on its own limited battery power. Eventually, a connection was rigged that allowed the battery in an uninterruptable power supply (UPS) unit to be charged from a car. The laptop's AC power supply could then draw AC power from the UPS unit to power the laptop and recharge the laptop battery. The failure of the original regional node laptop also killed our goal of uploading real-time CNAS weather information to the Air Force Weather Agency (AFWA) server from the drop zone using the Iridium satellite communication system.

Drop-zone redeployment

The original Talisman-Saber Exercise demonstration plans called for CNAS to be removed from the drop zone prior to airdrop operations (after collecting the 72 hours of observations). So, on June 18th, the drop-zone deployment was dismantled and the sensor agents transported to the Urban Operations Training Facility (UOTF) for deployment there (Figure 5). The UOTF deployment was nearly complete when, early in the morning of the 19th, CNAS was unexpectedly summoned back to the drop zone to provide real-time wind data during the airdrops. The UOTF deployment was dismantled and the sensor agents transported back to the drop zone. The CNAS team demonstrated a rapid re-

[9] A 90-minute drive over muddy "roads" from nightly sleeping facilities.

Figure 6: TACMET-Augmented Sensor Agent at UOTF (The device to the right of the sensor is not part of CNAS.)

sponse and set-up capability by having CNAS on-line and reporting observations within 15 minutes of arrival at the drop zone.[10]

UOTF deployment

After the drop-zone redeployment, CNAS was moved back to the UOTF. The UOTF is an urban environment that was constructed at the Shoalwater Bay Training Area. UOTF includes a number of buildings (commercial, retail, residential, shanty, rubble) built using both standard building materials and reconfigurable container-based structures. CNAS deployment at UOTF differed from the drop zone because the sensor agents were located much closer to each other and some of them were positioned on buildings that had different heights. In terms of WiFi communication, vegetation attenuation was replaced with urban reflection and interference. Fortunately, AC power for the replacement regional-node laptop was available at UOTF, eliminating that headache.

Several additional technical issues surfaced at the UOTF. Water from heavy rains entered the WIFI cable connectors and disrupted network operation. This was resolved on-site with application of petrolatum as sealant for the connections. A second issue resulted from periodic electromagnetic interference from a nearby demonstration system that was disrupting the 802.11b frequency range. This interference issue required no additional intervention, as the CNAS-agent software was sufficiently robust to recover gracefully during interference-free periods with no loss of data. CNAS pushed hourly weather observations, in properly-formatted structure, to the AFWA server as well as to the Australian Bureau of Meteorology (BOM) (Table 1).

A return to Australia in 2009?

The CNAS deployments at Talisman-Saber met all the technical goals and objectives established for the Exercise. These included: 1) automatic dissemination and posting of weather observations to AFWA and BOM weather servers, 2) support of the AFRL "COUNTER" small UAV demonstration, 3) support of airdrop operations, and 4) adapting to changing user requirements, observation needs, and mission objectives. Even in its experimental form, CNAS was deployed and providing information within 15 minutes of arrival at the drop-zone site—a highly visible achievement. As a result, CNAS has been invited to participate in the next Talisman-Saber Exercise in 2009.

[10]The sensor agents were positioned at the same locations as the original drop-zone deployment.

3. SILENCE WITH RESPONSIVENESS

Collecting sensor readings, aggregating them together at the cluster heads, and communicating them to console or regional nodes is well suited to the periodic communication windows used in CNAS. However, when more dynamic activities need to be performed (such as inspecting the local state of an agent, retasking an agent's activities or changing network policies, or having agents modify the world around them), having to wait until the next communication window can be an issue (as well as frustrating). Techniques are needed that improve network responsiveness without increasing the total amount of time that agents' radios need to be turned on.

Initially, CNAS agents were constrained to communicate using a "stock" OLSR routing protocol. Following the Talisman-Saber Exercise, this program-objective restriction was relaxed, allowing exploration of improvements to standard OLSR.

Persistent routing tables

One obvious improvement is to eliminate OLSR reinitialization at the start of each communication window. By having OLSR assume that no change has occurred while the wireless radio has been off, it can proceed when the radio is switched on using the state that existed at the end of the previous communication window. Intuitively, it is equivalent to having all changes happen at the moment the radio was switched back on. To the degree that the old routing information is reasonable, application-level communication is possible immediately and, hopefully, the cost of any adaptation is less than the loss of time required for complete reinitialization.

AFRL's Zenon Pryk explored maintaining OLSR routing information across communication cycles using a small CNAS agent network deployed indoors at AFRL. Experiments were run where a small utility which exercised all possible source & destination pairs. In this noisy indoor setting, OLSR stabilized in less than 15 seconds into the communication window versus the 40–50 seconds required by a from-scratch reinitialization. In a sparse, outdoor CNAS setting we expect the routing changes to be less dynamic due to the small number of paths available, and that OLSR with persistent routing tables will stabilize even faster.

Using organization knowledge in routing

MASs benefit greatly from an organization design [4, 6] that guides agents in determining when to communicate, how often, with whom, with what priority, and so on. However, this same organization knowledge is not utilized by general-purpose wireless network routing algorithms normally used to support agent communication. By making the routing algorithm aware of how agents interact in the CNAS organization, it can direct path exploration where it is most beneficial, find optimal paths faster, and conserve significant bandwidth for application use. General-purpose routing algorithms also treat all message as having the same priority. They spend the same amount of energy exploring paths

```
METAR KQRS 240755Z AUTO 11001KT 17/14 A3005 RMK PK WND 08003/52 SLPNO ESTMD ALSTG P1013;
METAR KQRS 240855Z AUTO 28000KT 15/14 A3007 RMK PK WND 29002/54 SLPNO ESTMD ALSTG P1014;
METAR KQRS 240955Z AUTO 24001KT 15/14 A3007 RMK PK WND 25003/54 SLPNO ESTMD ALSTG P1014;
METAR KQRS 241055Z AUTO 03000KT 14/13 A3008 RMK PK WND 35002/53 SLPNO ESTMD ALSTG P1014;
METAR KQRS 241155Z AUTO 18001KT 15/14 A3009 RMK PK WND 21004/50 SLPNO ESTMD ALSTG P1014;
METAR KQRS 241255Z AUTO 16000KT 15/14 A3007 RMK PK WND 22003/52 SLPNO ESTMD ALSTG P1014;
METAR KQRS 241355Z AUTO 19001KT 14/13 A3004 RMK PK WND 24003/54 SLPNO ESTMD ALSTG P1013;
METAR KQRS 241455Z AUTO 20001KT 14/13 A3006 RMK PK WND 22003/55 SLPNO ESTMD ALSTG P1014;
METAR KQRS 241555Z AUTO 19001KT 14/14 A3004 RMK PK WND 23003/50 SLPNO ESTMD ALSTG P1013;
METAR KQRS 241655Z AUTO 08001KT 15/14 A3001 RMK PK WND 10003/50 SLPNO ESTMD ALSTG P1012;
METAR KQRS 241755Z AUTO 22001KT 14/14 A3002 RMK PK WND 22005/52 SLPNO ESTMD ALSTG P1012;
```

Table 1: A portion of the METAR statements produced by CNAS at UOTF

for application messages where it is extremely important to find the best possible path, as they do exploring paths for messages that are not as important, and can be delayed. We wanted to make use of the organizational importance (priority) of various agent communications when controlling routing.

We modified a proactive routing protocol (CQRouting [7]) to incorporate knowledge of the CNAS organization (which agents communicate with which other agents given their roles and cluster membership, and the priority and frequency of the messages being sent). Using a detailed network-level simulation of CNAS, we observed a large reduction in the number of exploration messages relative to traditional routing protocols (such as OLSR). This resulted in a significant (43%) increase in the communication bandwidth available to the CNAS application. We also obtained an increased global utility with fewer exploration messages by having the eCQRouting routing protocol bias path exploration based on message priority characteristics of the CNAS organization. This work is detailed in Zafar et. al. [11].

Rapid radio cycling

Maintaining routing information across CNAS communication cycles and providing organizational communication information to the routing algorithms allow application-level communication to be performed using shorter communication windows. Shorter windows allow more of them (a higher communication-cycle frequency) to be supported with the same power expenditure. At the extreme, we would like to have each agent turn on its radio and, as quickly as possible, determine if it needs to be available to support any message communication during that window. If no communication support is needed (either to send a message, receive a message, or perform multi-hop forwarding), then the agent can turn off its radio straight away. The shorter these "turn on and determine" windows can be made, the more frequently they can be cycled while maintaining the same level of energy expenditure. Frequent cycles mean more CNAS responsiveness.

Intuitively, we could estimate the time needed for any agent to get an "I need to communicate" broadcast message to all other agents in the network. Then, when each agent begins a cycle, it keeps its radio turned on for that long to hear if any agent (including console and regional nodes) wants to communicate. If no such message is received during that time, it turns off its radio.

For a small network, this broadcast time will be short, but the time increases as the number of hops grows. However, for very brief "no message" communication windows, the time until the next window is also short, so the need to communicate a message across the network in one cycle is lessened. If the message does not make it all the way to its destination in the current window, it will be stored and forwarded in the next. Thus, what we really seek is having the agents along a segment of the message path determine that they are needed in the current cycle. The goal is obtaining "waves" of communication-forwarding windows appearing where and when needed in the network. As with the organizationally based routing work described previously, CNAS can benefit by having the network-routing layer be aware of the application-level communication window and cycles. Making the "no message" decision at a lower communication layer can be very efficient.

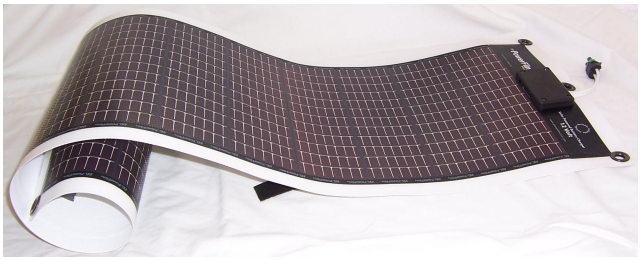


Figure 7: Rollable Solar Panel

We are beginning development and evaluation of such a rapid-cycle, communication-window-aware routing protocol for CNAS-like networks. What differentiates this work from recent power-aware routing research is that we assume all agents' radios remain off (except for the frequent "any communication?" windows) until packets along a path need to be propagated. Rapid cycling is ideal for conserving communication energy in low-bandwidth, highly responsive settings.

4. SOLAR HARVESTING

Sensor agents whose power reserves can be expected to be replenished from time to time add new challenges in power management. We are currently exploring the addition of a rollable (thin film on plastic) solar panel (see Figure 7) to each sensor agent that allows its battery reserves to grow (up to full battery capacity) if the agent is in an unshaded location and the sun is shining. The activity decisions that are made by agents with replenishable resources now must consider how much additional power may become available and when. We assume that the CNAS sensor network is provided with the sunshine forecast (from sources outside the network), but each agent needs to learn the implications of the forecast on its own power reserves. A sensor agent that recognizes that it is shaded by a hill in the afternoon should realize that it cannot expect to obtain much power replacement if the forecast is for cloudy weather until noon. On the other hand, an unshaded agent could anticipate being able to perform additional power-intensive activities, based

on its expectation of power replenishment.

A requirement of CNAS is that the positioning of sensor agents cannot be optimized for either data collection or solar harvesting. Eventually sensor agents may be airdropped (rather than hastily deployed by hand). In either case, the exact placement of sensor cannot be regulated, which requires the shade and tilt attenuation to be determined once the agents are situated. Thus, agents in CNAS, like agents deployed in other real-world applications, need to determine aspects of their local environments in order to make appropriate decisions. For predicting solar-harvesting amounts, each agent must develop a model of the efficiency of its solar panel, attenuation due to positioning of the panel relative to the sun, attenuation due to shading (by trees, hills, buildings, etc.), and attenuation due to cloud cover. With such a model, an agent can translate a temporal sunshine forecast into a realistic estimate for the solar energy to be harvested at its location. This model development must be done quickly, as power-management decisions will be inaccurate until each agent becomes aware of its surroundings.

We have developed a strategy of factoring solar-harvesting model development (which would typically be done using multi-agent learning) into two phases: pre-deployment (site independent, individual agent) learning and post-deployment (site dependent, multi-agent) model completion. By performing as much site-independent learning as possible prior to deployment, we simplify what needs to be determined on-site by each sensor agent. Furthermore, we use collective sharing of local-observation information, in conjunction with temporal and spatial constraints in relating this information, to reduce the number of observations needed to perform each agent's model-completion activities. In all but the most unlikely of environmental conditions, our two-phased strategy allows individual-agent harvesting models to be completed using only the first and second day's observations. Thus each agent can make fully informed solar-harvesting predictions given cloud forecasts starting on the third day. This work is detailed in Zafar and Corkill [10].

5. NEXT-GENERATION HARDWARE

The PASTA microsensor platform used in CNAS is already showing its age. New PASTA "Ziti" processors are no longer available and, as discussed above, problems interfacing the Crossbow to the PASTA remain unresolved. Similarly, the Netgear MA111 (IEEE 802.11b) USB Wireless adapter used in CNAS is outdated. After participating in the Talisman-Saber Exercise, we began looking into new hardware possibilities for CNAS agents.

One promising line of replacement processors are the PXA255-based Connex and the PXA270-based Verdex motherboards from Gumstix, Inc. These have become very popular, and they can be interfaced with compatible sensor, WiFi, and GPS hardware. The Gumstix have the advantage of being commodity devices readily available at commodity prices. However, they have the disadvantage of not being designed or targeted for low-energy consumption. The PASTA has a distributed-peer architecture that can decouple processing from peripheral operation, allowing even the central processor to be powered down to lower total system-power expenditure [8]. For use as a CNAS agent, however, this limitation is not a serious liability. Managing the high-expenditure WiFi and GPS devices has been the prime focus of our CNAS effort, and having the Gumstix running continually is acceptable.

Porting CLISP to the Gumstix

GBBopen and its Common Lisp substrate provide many advantages for CNAS. One important advantage is the ability to compile source code directly in the Common Lisp image, without the need for a traditional development environment on the CNAS agent. We used the open-source Common Lisp implementation, CLISP,[11] in the PASTA-based sensor agents, so also providing CLISP on the Gumstix processors was crucial. An important advantage of CLISP is that it is structured as a small C-based kernel that operates in conjunction with a platform-independent byte-code compiler and virtual-machine executer. A major disadvantage of CLISP, however, is that it does not support Lisp multiprocessing (process threads), which complicates real-time event processing. However, the small and portable C-based kernel and compact byte-code executer are well suited to the memory space available on the PASTA and Gumstix processors.

On the PASTA, we were able to make use of the Debian ARM packaging of CLISP 2.34 performed by Will Newton. The Debian package allowed us to bypass cross compiling the CLISP kernel. This was not the case for the Gumstix, as the kernel and library code there are tightly coupled with the buildroot cross-compilation toolchain. CLISP has a fairly complicated build process that has evolved over the years to support deployment on a wide range of hardware and operating system platforms. At one point, cross compilation of CLISP was supported, but that capability has long passed into disrepair. We did not have the space or desire to use an ARM-based GNU development environment on the Gumstix itself, so we had no alternative but to explore CLISP cross compilation.

Building CLISP involves the following eight steps:

1. `make init` — prepares all symbolic links and utilities
2. `make allc` — makes all `*.c` files
3. `make lisp.run` — makes the C-kernel executable
4. `make interpreted.mem` — creates a memory image with everything uncompiled
5. `make halfcompiled.mem` — creates a memory image with `compiler.fas` and the rest uncompiled
6. `make lispinit.mem` — makes all `*.fas` files and creates a memory image with everything compiled
7. `make modular` — makes the base module set
8. `make install` — completes the CLISP installation

We developed a procedure for performing the first three steps (with manual interventions in the automated build scripts), to create a `lisp.run` executable for the Gumstix using the Linux/x86 buildroot cross-compilation toolkit. Compilation of the Common Lisp source files for the CLISP compiler and the rest of the CLISP Common Lisp environment was performed using a normal CLISP image running on the Linux/X86 development platform with appropriate compilation settings for the ARM-based deployment.[12] Then the compiled output files (`*.fas` files) and the `lisp.run` executable are copied to the Gumstix and the `lispinit.mem` memory image is built there. Again, this requires manual intervention, however the result is the latest CLISP release running smoothly on the Gumstix.

[11] http://clisp.cons.org/

[12] With the exception of calls to some low-level routines, CLISP's byte-code compilation is platform independent. The compiler settings address these low-level issues.

Given the suitability of CLISP for small, embeddable devices such as the Gumstix, restoring cross-compilation capability to the automated build scripts would eliminate the labor-intensive manual build process that we were forced to use. As an open-source project with a large user base, non-trivial changes to the current build process will require care and perseverance in order to be accepted. So, at least for now, recompiling CLISP for new system-software updates (such as the new Gumstix OpenEmbedded (OE) build process) will remain labor intensive.

We are only beginning to consider sensor hardware for the Gumstix platforms. Given the growing success of this marketplace, however, the CNAS goal of exploring hardware capabilities that are likely to become cost-effective for production deployments in the next few years is quickly becoming a reality.

6. SUMMARY

CNAS has been successful as both a deployed agent-based sensor network and as a research environment. Working on CNAS has also been a lot of fun! Sensor agents operating near the limit of radio-communication range with their radios turned off most of the time present different challenges than traditional MAS settings. Our emphasis to date has been on achieving basic CNAS capabilities and reliability, and we are only starting to expand our use of blackboard-system capabilities in incorporating more complex activities and adaptive reasoning into CNAS.

CNAS has become acceptably reliable for use in the field (literally), and there is interest in demonstrating CNAS on board ships at sea. The latest hardware directions, and accompanying low cost, will also enable permanent deployment of CNAS agents in areas where the economic cost of losing of a sensor agent due to damage or theft becomes tolerable. The future for agent-based sensor networks like CNAS is growing ever brighter. We are already looking forward to Talisman-Saber 2009!

Acknowledgments

Kevin Bartlett, Robert Fleishauer, Walter Koziarz, Wenchian Lee, Zenon Pryk, Wilmar Sifre, Lok-Kwong Yan, and Paul Yaworsky of AFRL/IF and Huzaifa Zafar of UMass contributed to CNAS.[13] Walter Koziarz, Zenon Pryk, Lok-Kwong Yan. Wilmar Sifre, and Robert Fleishauer supported the Talisman-Saber deployments of CNAS in Queensland. Carrie Kindler of ITT Industries developed the map-based 2D graphic display client used at the regional and console nodes. Steven Hoffman performed much of the tedious work of cross-compiling the CLISP `lisp.run` C-kernel for the Gumstix. Douglas Holzhauer (now retired from AFRL and teaching at SUNYIT) initiated the CNAS project and directed it through its formative months and PATRIOT 2006 deployment.

7. REFERENCES

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3(3):325–349, 2005.

[2] D. D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, Sept. 1991.

[3] D. D. Corkill, D. Holzhauer, and W. Koziarz. Turn off your radios! Environmental monitoring using power-constrained sensor agents. In *Proceedings of the First International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, pages 31–38, Honolulu, Hawaii, May 2007.

[4] D. D. Corkill and V. R. Lesser. The use of meta-level control for coordination in a distributed problem-solving network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, Karlsruhe, Federal Republic of Germany, Aug. 1983. (Also published in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986.).

[5] R. S. Engelmore and A. Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

[6] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 2005.

[7] S. Kumar. Confidence based dual reinforcement Q-routing: An on-line adaptive network routing algorithm. Master's thesis, University of Texas at Austin, Austin, Texas, 1998.

[8] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with >1000X dynamic power range. In *Information Processing in Sensor Networks (ISPN 05), special track on Platform Tools and Design Methods for Network Embedded Sensors*, Los Angeles, California, Apr. 2005.

[9] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 70–84, Rome, Italy, July 2001.

[10] H. Zafar and D. Corkill. Simplifying solar-harvesting model development in situated agents using pre-deployment learning and information sharing. In *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks (ATSN-08)*, Estoril, Portugal, May 2008. To appear.

[11] H. Zafar, V. Lesser, D. Corkill, and D. Ganesan. Using organization knowledge to improve routing performance in wireless multi-agent networks. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, Estoril, Portugal, May 2008. To appear.

[13] Professors Victor Lesser and Deepak Ganesan advised Huzaifa on the organization-based routing research.

E Organizational Routing: The AAMAS-08 Paper

This appendix contains a preprint of the AAMAS-08 paper: "Using Organization Knowledge to Improve Routing Performance in Wireless Multi-Agent Networks," by Huzaifa Zafar, Victor Lesser, Daniel Corkill, and Deepak Ganesan. This paper will appear in the *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS-08), Estoril, Portugal in May 2008.

Using Organization Knowledge to Improve Routing Performance in Wireless Multi-Agent Networks

Huzaifa Zafar, Victor Lesser, Daniel Corkill, Deepak Ganesan
Department of Computer Science
University of Massachusetts Amherst
{hzafar, lesser, corkill, ganesan}@cs.umass.edu

ABSTRACT

Multi-agent systems benefit greatly from an organization design that guides agents in determining when to communicate, how often, with whom, with what priority, and so on. However, this same organization knowledge is not utilized by general-purpose wireless network routing algorithms normally used to support agent communication. We show that incorporating organization knowledge (otherwise available only to the application layer) in the network-layer routing algorithm increases bandwidth available at the application layer by as much as 35 percent. This increased bandwidth is especially important in communication-intensive application settings, such as agent-based sensor networks, where node failures and link dynamics make providing sufficient inter-agent communication especially challenging.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing Protocols; C.2.4 [**Distributed Systems**]: Distributed Applications; I.2.11 [**Distributed Artificial Intelligence**]: Multi-Agent Systems

General Terms

Algorithms, Design, Experimentation, Verification

Keywords

Agents, Multi-Agent Sensor Networks, Organization Design, Wireless Routing, Communication, Bandwidth

This work is supported in part by the AFRL "Advanced Computing Architecture" program, under contract FA8750-05-1-0039. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views contained in this paper are the authors.'
This work was also supported by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.
Cite as: Using Organization Knowledge to Improve Routing Performance in Wireless Multi-Agent Networks, Zafar H., Lesser V., Corkill D., and Ganesan D., *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.
Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

In any multi-agent system (MAS), there is a cost in communicating among agents in the form of time delay and bandwidth expenditure. In wireless networks, agents are often multiple communication hops away from one another. Supporting multi-hop communication involves an additional network-layer exploration cost in determining communication paths to these agents. This exploration cost becomes significant as the size of the network increases (Table 1) and as network stability decreases. Exploration expenditures reduce the effective bandwidth available to the application, resulting in additional time required in transferring the same amount of application data from one agent to another.

Size of Grid	OLSR		DSDV	
	Total	Per Agent	Total	Per Agent
9 agents	1038	115.33	178	19.77
16 agents	3880	242.5	521	32.56
25 agents	9401	376.04	1011	40.44
36 agents	17535	487.08	1654	45.94
49 agents	32353	660.26	2136	43.59

Table 1: Exploration costs increase with size of the network. We use a wireless ad hoc grid network. Each agent lies on the edge of the transmission range of its neighboring agents, allowing for only 4 neighbors per agent. The table shows the number of exploration messages sent in the first 60 seconds for OLSR (a standard routing algorithm) and for DSDV (a standard proactive distance-vector-based routing protocol)

CNAS (Collaborative Network for Atmospheric Sensing) [3] is one such wireless MAS application. It is a collection of sensor agents, spread sparsely over a geographical region. CNAS makes an overall assessment of detailed local atmospheric conditions by combining data from sensor agents within predefined cluster areas. A hierarchy of communication is used, where all the sensor agents collect data from their environment and transfer it to a dynamically designated sensor agent performing a second role of a *cluster head*. The cluster head is responsible for aggregating and reporting the conditions in its cluster area. Moreover, the cluster head is also responsible for providing direction to the rest of the agents in its cluster. Due to network dynamics such as agent and link failures and changes in communication-link characteristics, the network-level routing algorithm at each agent in the network performs regular exploration of links to its neighbors. This exploration determines neighbor availabil-

ity and the utility of the links with them. Agents acting as cluster heads communicate with a *regional agent*, but with less frequency. Nevertheless, paths to the regional agent must also be maintained. However, regional-agent communications are of extremely high priority and there are greater consequences for delayed delivery.

MAS applications use organization design to determine when to communicate, how often, with whom, with what priority, and so on [4, 5]. By being aware of which agents are interested in communicating with whom at the network level, we are able to direct network-layer exploration where it is most beneficial, find optimal paths faster, and conserve significant bandwidth for application use. We also make sure of the organizational importance (priority) of various agent communication. General-purpose routing algorithms treat each message as having the same priority. These algorithms spend the same amount of energy exploring paths for application messages where it is extremely important to find the best possible path, as they do exploring paths for messages that are not as important, and can be delayed or routed less directly. By using the organizational priority of communication, we are able to improve application performance with fewer exploration messages.

We modified a pre-existing proactive routing protocol (CQRouting [8]) to incorporate the organization knowledge that specifies for each agent whom to communicate with, the priority of the message being sent, and the frequency with which the communication happens. Using a detailed network-level simulation of CNAS we obtain a large reduction in the number of exploration messages relative to traditional routing protocols. This results in a significant increase in the communication bandwidth available to the CNAS application.

2. RELATED WORK

Developing a path to a destination agent is performed by the underlying routing algorithm, and there has been considerable research done in developing general-purpose routing algorithms [1, 10, 11, 12]. However all proactive routing algorithms (where the routing table is developed before sending application-level messages) develop paths to every node in the network, irrespective of the need for or importance of the path. This results in significant discovery and maintenance cost on the part of the routing algorithm due to the assumption that every node wants to communicate with every other node in an extremely dynamic network. Moreover, routing algorithms explore more frequently as the dynamics of the network increase. On the other hand, reactive routing algorithms suffer from an added exploration delay when sending data messages (on top of the delivery delay) and are not suitable for highly responsive agent networks.

2.1 OLSR

OLSR (Optimized Link State Routing protocol) [2] is the most commonly used proactive routing protocol in wireless sensor networks. OLSR is optimized for large, dense, wireless ad-hoc networks, and computes optimal forwarding paths by flooding exploration messages to all nodes in the network. Exploration in OLSR happens in two stages. The first stage performs a distributed election of a set of multi-point relay (MPR) nodes that are solely responsible for forwarding messages within the network. The MPR set has the property that each node in the network is a direct neighbor of an MPR, and that two MPRs are at most separated by a single node. The second stage in OLSR develops a routing table at each node that defines the next hop on the path to every other node in the network. This is done by having each node flood the network with topology control (TC) messages, and using other TC messages in generating its local routing table. By using only the MPR nodes to propagate exploration messages, OLSR reduces some of the redundancy of the flooding process. OLSR uses *destination-based exploration*, where a node develops a table entry for a destination only after it receives a TC message from that destination.

2.2 CQRouting

CQRouting [8] is an extension to the QRouting [1] protocol based on the distributed QLearning algorithm [13]. The network is constructed from a distributed collection of learners. A QValue, $Q_x(y, d)$, for a node, x, is the expected time taken to transfer a message to destination node, d, through neighbor node, y. A policy determines which neighbor a message is forwarded to so that it reaches its destination with minimum delay.

QRouting uses the "Full Echo" algorithm to perform exploration [1]. Each node in the network periodically requests the policy being used by each of its neighbors in determining paths to various nodes in the network (which is determined by the routing table of that neighbor, generated using the Bellman-Ford Algorithm for routing, first described in the ARPANET [9]). The querying node then updates its own tables based upon the following QLearning equation:

$$Q_x(y, d) = Q_x(y, d) + \alpha(Q_y(\hat{z}, d) + q_y - Q_x(y, d)) \qquad (1)$$

where α is the learning rate that regulates the effect of previous QValues on the current one. $Q_y(\hat{z}, d)$ is the delay in sending a message from node y, through y's best neighbor $\hat{z}$, to node d. q_y is the delay in sending a message from node x to node y.

In CQRouting each node also maintains a confidence in its QValues, and shares this confidence with other nodes in the network along with its routing tables. The closer the confidence value is to 1, the better the node expects the QValue to reflect the state of the network. When an agent updates its QValues from the policies received from its neighbors, the agent uses the confidence its neighbor reports in determining its own QValue as follows

$$Q_x(y, d) = Q_x(y, d) + \alpha C_x(y, d)(Q_y(\hat{z}, d) + q_y - Q_x(y, d)) \quad (2)$$

For each time step that the agent does not explore, its confidence decays due to the expectation that the network is dynamic and its QValues might not reflect the current state of the network.

3. eCQROUTING

We develop a new wireless routing protocol called eCQRouting, that combines and extends OLSR and CQRouting. OLSR pays a higher exploration penalty due to the communication required in setting up the MPR nodes and in developing nodes' routing tables. In eCQRouting, organization knowledge is used to determine the explorer nodes. The rest of the nodes in the network take advantage of these exploration messages in developing their own routing tables by eavesdropping on the exploration messages and their results. Secondly, OLSR uses frequent network flooding to keep the

routing tables synchronized with fluctuations in the network. Incorporating CQRouting with OLSR smooths the effect of network fluctuations, reducing the need for frequent flooding of exploration messages. Furthermore, by reducing flooding, the benefit of electing MPRs is also reduced. This reduction is enough that it does not justify the election cost of MPRs, so we remove MPR development from eCQRouting. The following subsections describe additions to the eCQRouting protocol to allow for a smoother implementation in wireless sensor networks.

3.1 Damping Factor

Use of a learning factor (α) in QLearning, introduces the possibility of the utility of agent A dropping below the utility of agent B, even though B is being used as its next hop. Since we are interested in minimizing QValues, whenever A sends its policy to B, B will switch to using A as its next hop due to the lower utility. By performing multiple explorations over that path, QLearning eventually overcomes this issue and converges to the corrects QValues. However, because every exploration message consumes potential application-level bandwidth, we modified the QRouting algorithm to dynamically adjust the learning factor each time an agent updates its QValues. The new learning factor depends on the current QValue and the QValue provided by the agent's neighbor. This allows for better and faster convergence of QValues with fewer explorations while retaining the benefit of the learning factor in keeping the optimal path from flip-flopping between two close paths. The learning factor regulates how closely the current QValue reflects the "present" delay to the destination verses the values calculated in the "past." This implies that the closer α is to being equal to one, the stronger the guarantee that the QValue of any agent to its destination is greater than the QValue of its next hop to the destination. From the QLearning algorithm:

$$\begin{aligned} Q_x(y,d) + \alpha(Q_y(\hat{z},d) + q_y - Q_x(y,d)) \quad &> \quad Q_y(\hat{z},d) \\ \alpha > \frac{Q_y(\hat{z},d) - Q_x(y,d)}{Q_y(\hat{z},d) + q_y - Q_x(y,d)} \end{aligned} \tag{3}$$

The above value gives us a lower bound on the value of α, with the upper bound being 1. The new damping factor is:

$$\alpha = 1 - \hat{\alpha} * (1 - \frac{Q_y(\hat{z},d) - Q_x(y,d)}{Q_y(\hat{z},d) + q_y - Q_x(y,d)}) \tag{4}$$

By changing $\hat{\alpha}$, we are able to regulate how close we want our α value to be to 1, but at the same time avoid damping issues. For eCQRouting, we modified the α formula as follows

$$\alpha = 1 - \hat{\alpha} * (1 - \frac{Q_y(\hat{z},d) - Q_x(y,d)}{C_x(y,d) * (Q_y(\hat{z},d) + q_y - Q_x(y,d))}) \tag{5}$$

4. ORGANIZATION AND ROUTING

We next describe incorporating organization knowledge in the routing algorithm used at each agent and the corresponding routing-level exploration changes, given this knowledge.

4.1 Knowledge of the organization

The direction and priority of communication between each agent role is represented as a weighted graph. A local copy of the graph is to used to make exploration decisions based on the roles an agent is assigned. Figure 1a shows the CNAS organization. The corresponding communication graph is depicted in Figure 1b.

Figure 2a shows a second, more complicated organization as used in CASA (Collaborative Adaptive Sensing of the Atmosphere) [7] another agent-based atmospheric sensor network with considerably higher bandwidth requirements than CNAS. In CASA, there are four roles an agent can take; Rad(s) (Radar(s)), FD (Feature Detector), FR (Feature Repository) and Opt (Optimizer). The organization is hierarchical. A CASA agent is responsible for communicating with other neighboring agents that are performing the same role as itself as well as communicating with parent agents, however with a lower priority. Also, agents with roles higher on the hierarchy communicate less frequently, and these messages have a much higher priority than communications among agents that are lower in the hierarchy. The corresponding graph communication graph with priorities is shown in Figure 2b.

Each agent is also provided with a description of all the other agents in the network (that are part of the organization), the roles they play, and the destination agents they communicate with. We believe that providing this kind of "global" view of the organization knowledge is reasonable, as it is considerably more stable than the routing tables used by the network-layer routing algorithms and this global perspective has negligible costs when compared to sharing routing tables. As the size of the organization increases, the scale of both the organization structure and the routing table increases. However, the organizational knowledge limits the size of the routing table by guiding exploration messages (and consecutively path development) to only those destination agents reflected in the organization knowledge.

4.1.1 Exploration Based on Organization

A destination agent in the organization knows (from its organizational knowledge) all the agents that communicate with it. Also, in most networks like CNAS, the number of destination agents is much smaller than the number of source agents. In eCQRouting, the destination agent performs periodic exploration of the network[1]. The extent and direction of an agent's exploration is based on the placement of agents that communicate with it. The initial exploration messages are used by the destination agent to find the other agents. Future exploration messages are then directed according to the location of agents that communicate with the destination. The QValue at the source agent is defined by the delay in receiving the exploration messages.

4.1.2 Exploration based on Confidences in QValues

In eCQRouting, an agent explores when its confidence drops below a certain threshold. Since exploration happens at the destination, yet confidence is calculated at the source, confidence is piggybacked on application messages from the source to its destination. The destination agent updates a local copy of the confidences of all agents that send messages to it and uses its local copy to determine when to explore. Furthermore the destination agent confidences decay over time so it will explore even if there are no application messages sent to it.

[1]We assume communication links between nodes are bi-directional and symmetrical. Even though this is not true for most wireless sensor networks, both OLSR and CQRouting assume the same.

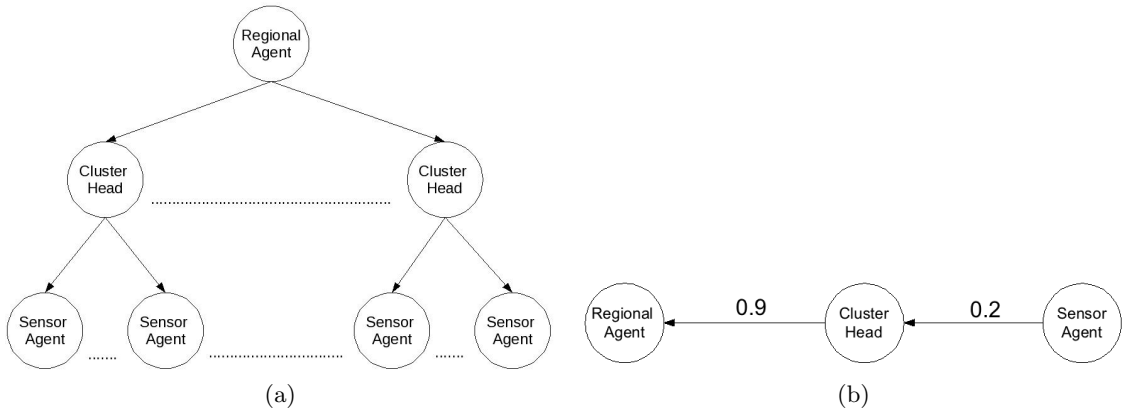


Figure 1: (a) CNAS Organizational Design and (b) CNAS communication directions and priorities

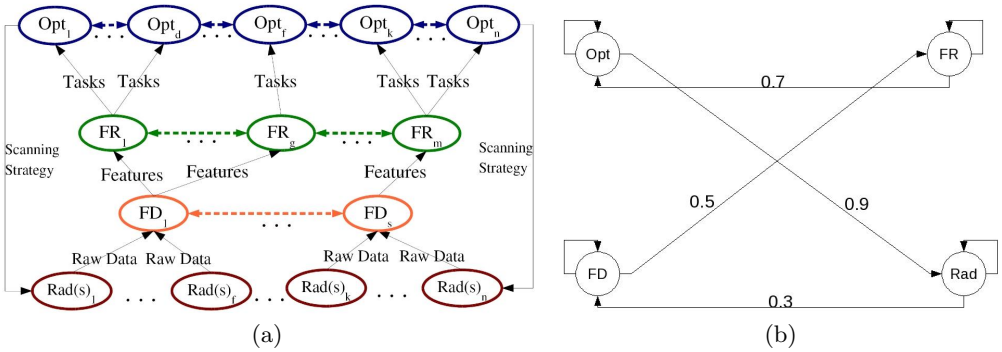


Figure 2: (a) CASA Organizational Design and (b) CASA communication directions and priorities

4.2 Message Priority

In Section 4.1, we defined our organization knowledge as a weighted graph, where the weight on an edge of the graph is based on the priority of the messages sent from one agent to another. Messages that are critical have a much higher priority as compared to other messages and require more exploration in order to maintain the optimal path, even though the number of messages sent might be much lower. For example, the "Optimizer" role in CASA includes setting the scanning strategy for the "Radar" agents. Lost scanning strategy messages can be very detrimental to the application as it could potentially mean radar agents are following an out-dated scanning strategy. CASA uses focused radars, so this could lead to loss of tracking data.

At the application level, an "Optimizer" agent can delay sending messages to other "Optimizer" agents if it has a scanning strategy that needs to be communicated. Each agent uses a priority queue which regulates when to send what message based on its importance. A problem with this strategy is priority-based starvation, where low priority messages wait infinitely while high priority messages are added ahead of them in the queue. As a routing algorithm however, we would like lower priority messages to be communicated using suboptimal paths and be eventually delivered, while saving optimal paths for high priority messages.

In eCQRouting we solve this problem by using two techniques. The first technique uses an approximate measure of message priority for each source-destination pair. The priority is then used to regulate the frequency with which exploration is performed by the destination agent. In the second technique, each agent has a local measure of the performance of the MAS based on the probability of dropping a message to the destination. For each message, decisions are then taken based on this local measure.

4.2.1 *Exploration based on message priority*

Consider a simple case shown in Figure 3. After the first

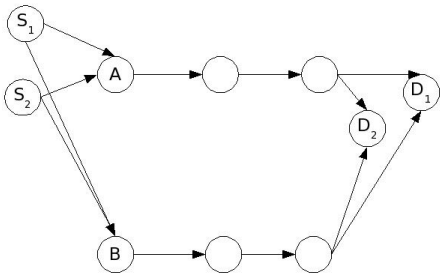


Figure 3: Priority behavior in a simple network. $S_1 \rightarrow D_1$ has a much higher communication priority than $S_2 \rightarrow D_2$

round of exploration is completed, S_1 and S_2 both deter-

mine A to be the best path from themselves to D_1 and D_2 respectively. However as both of them start using A as their next hop, the network discovers the bandwidth from A to D_1 and D_2 is insufficient to transmit both sets of messages, resulting in an increased time delay. The next time an exploration happens, both D_1 and D_2 explore, find the path through B to be the better next hop and S_1 and S_2 both choose B causing an overload on B.

In eCQRouting however, we lower the threshold at which a destination agent explores based on the priority using the formula $threshold = priority * sd\text{-}threshold$. The sd-threshold is uniform for all source-destination pairs. This tolerates a lower variance on the value of high priority messages before forcing exploration to resolve them, and results in better mean and variance development for those values. It also means that if S_1 has a higher priority over S_2, it will explore more, forcing S_1 to pick a suboptimal path through B, in order to make sure that S_1 resolves the bottleneck. The next time S_2 explores, it would keep the optimal path through A. The technique is suboptimal is a static network, but in a dynamic network permits a high priority source-destination pair to be more aware of the state of the network in developing paths to the destination.

The disadvantage of this algorithm is the system has to wait for the next exploration message to determine new paths, which delay decisions.

4.2.2 Exploration based on message loss

The second technique takes into account the expected effect of performance relative to message loss and performs local decisions based on that effect. Suppose S_1 and S_2 start sending messages and realize 10% of their messages are being dropped before they reach A. If the performance curve (see Figure 10a) for messages from S_1 is such that S_1 cannot handle the 10% loss, S_1 will pick the path through B with the expectation that that path will deliver more than 90% of its messages.

As agents explore, link probabilities are shared in the following way; assume that the network in Figure 3 has the probabilities shown in Figure 4. Here, E will share

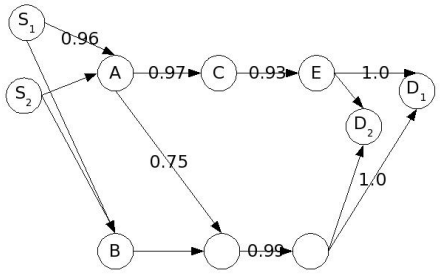


Figure 4: A simple network with probability of sending a message from one agent to another

1.0 with agent C. Agent C will share $0.93 * 1.0 = 0.93$ with A. A has two options: $0.75 * 0.99 * 1.0 = 0.7425$ and $0.97 * 0.93 = 0.9021$. Since 0.9021 is greater, A shares that value with S_1. In making a decision S_1 will either pick A (knowing the probability of messages to get through to be 0.866) or pick B (assuming the probability of getting the message across is 1).

The disadvantage of this technique is it takes the next exploration to make a global decision and a local decision might not always be optimal. The second disadvantage is that it takes time to learn the probability of dropping messages on any link.

5. EXPERIMENTAL ANALYSIS

All experiments were conducted using NS2 [6], a standard network simulator. NS2 provides models for standard network link dynamics as well as message interference, both of which strongly govern available bandwidth. We also used the widely used NS2 implementation of OLSR developed at University of Murcia.[2] In our experimental analysis, we evaluate the benefit of adding organization knowledge to the lower level routing algorithm (eCQRouting). We measure: 1) the effect on performance with increasing number of exploration messages, 2) the scalability of the algorithm as the number of non-application agents increase, 3) the scalability of the algorithm as the density of agents increase, and 4) the effect of bandwidth available on messages with different priority.

5.1 The effect of exploration on performance

This experiment used the CNAS organization structure on the 1-D network shown in Figure 5.

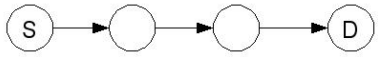


Figure 5: 1D network

Here S represents a sensor agent and D a cluster head agent. Next, neighbors that are not used by the high-level application and thus not part of the organization structure, are added to both the sensor agent and the cluster head in order to increase the number of exploration messages on the path from S to D. The neighbors are added such that they do not add additional paths from the source to the destination (Figure 6 shows an example network after adding 3 neighbors).

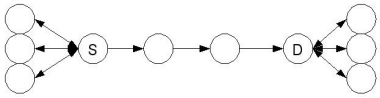


Figure 6: Adding 3 neighbors to both source and destination in Figure 5

With no additional neighbors, the maximum bandwidth available if no exploration were to be done was determined to be 180Kbits/sec. Agent S sends 180 1Kbit messages to agent D every second for 150 seconds. The average number of messages that reach agent D every second determines the bandwidth. We performed 10 runs and averaged the bandwidth over the 10 runs. The network is completely stable: no links fail during the course of the experiment. As additional neighbor agents are added to the network, traditional

[2]http://masimum.dif.um.es/?Software:UM-OLSR

routing algorithms (OLSR and DSDV[3] in our experiments) use additional exploration messages in order to include these agents in the routing tables of all other agents in the network. eCQRouting prevents this from happening by taking advantage of the CNAS organization knowledge that specifies that agent D to be the only destination agent in the network. The effect on the bandwidth available to agent S in sending its data to agent D is shown in Figure 7

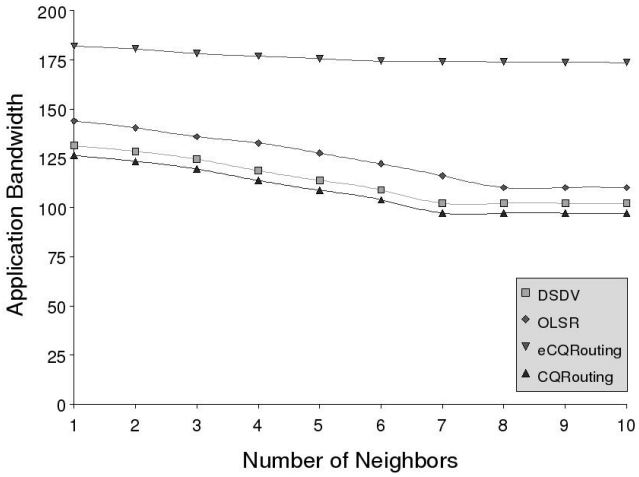


Figure 7: Changes in bandwidth as the number of neighbors (and in-effect the number of exploration messages) increases. Bandwidth is calculated in KBits/sec and averaged over 150 seconds

As the number of neighbors increases, the application level bandwidth available when using eCQRouting is significantly better than the next best algorithm (OLSR), with performance improvements of 20.9% with one neighbor to 36.55% with 10 neighbors. The algorithm also provides 30.5% additional bandwidth with one neighbor over CQRouting, which increases to 44% improvement with 10 neighbors. The reason for this improvement is the reduced number of exploration messages in eCQRouting. eCQRouting uses 15% of the number of exploration messages in OLSR with one neighbor, which drops to 7% with 10 neighbors.

5.2 Scalability in number of agents

We again use the CNAS organization, however the agents are now randomly placed within a predefined area. Networks of size 4 through 100 were evaluated. The density of the network remains the same throughout. Three sensor agents and one cluster are head placed randomly along with other agents as needed. The network is no longer stable: a link between two agents fails randomly every second with a probability of 0.2.

As the size of the network increases, the number of exploration messages also increases. However, since the density of the network remains the same, the number of hops to the destination also increases. The algorithm needs to be able to generate paths that are viable over the longer distances. Each sensor agent sends 180 1Kbit messages to the cluster head agent every second for 150 seconds. We measure the average number of messages received by the cluster head every second; which defines the application-level bandwidth for the run. Figure 8 shows the effect of this increasing size on the bandwidth averaged over 10 runs.

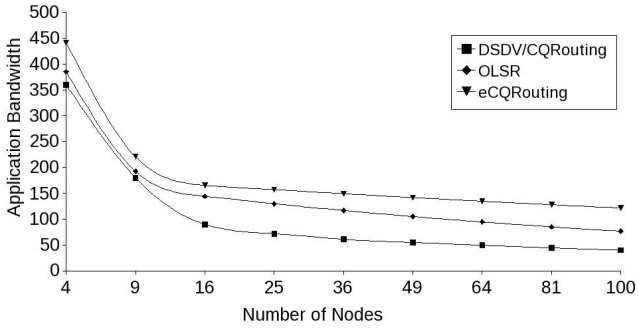


Figure 8: Changes in bandwidth as size of the grid network increases. Bandwidth is the average bandwidth over 150 seconds, measured in KBits/sec

Figure 8 shows an initial drop in the bandwidth available, as the destination agent is no longer 1 hop away from the sensor agents. The reduction in application-level bandwidth diminishes as additional hops are added between the source and the destination. The performance improvement in this experimental setting of eCQRouting over OLSR range from 10% for the 4 agent network to 35% for the 100 agent network. The improvement over CQRouting goes from 16.5% for the 4 agent network to 66% for the 100 agent network. Both DSDV and CQRouting have similar performance due to similar source-based exploration algorithms. As the number of hops increase, the time to converge to the optimal path also increases.

5.3 Scalability in agent density

We start with the 25 agent CNAS layout defined in the previous section. We keep the area constant, and add 25 to 200 agents at random positions within the network. This increases the density of the network, which in turn increases the number of paths from the source agents to the destination agent. Figure 9 shows the average application bandwidth of 20 runs for each density variation.

Figure 9 shows eCQRouting performs about 17% better than OLSR in the 25 agent network with minimal density. When density is doubled, the performance improvement increases to 19%. At triple the density, the performance improvement is 10%. When the density reaches 8 times the minimal density, OLSR outperforms eCQRouting. This is because the benefit of building MPR agents outweigh its costs at this density, allowing OLSR to do much better than all other routing algorithms.

5.4 Effect of the bandwidth available on messages with different priority

To explore the effect of priority, we used the CASA organization with priorities on messages as shown in Figure 2b. In particular, we look at networks where the MAS can tolerate the loss of a certain percentage of messages with-

[3]DSDV is a table driven routing algorithm for adhoc wireless networks based on the Bellman Ford Algorithm. Its similar to CQRouting in that agents share routing tables (rather than sharing QValues) with their neighbors and use the shared tables in developing local routing algorithms.

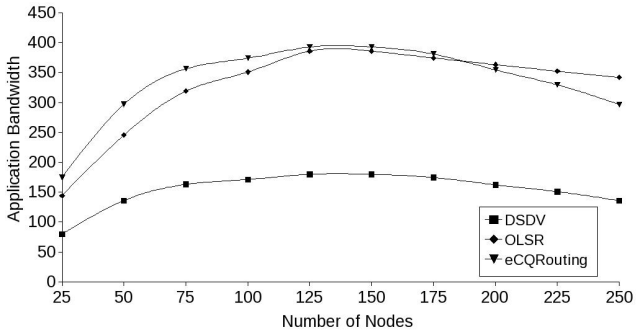


Figure 9: Changes in bandwidth as network density increases. Bandwidth is the average bandwidth over 150 seconds, measured in KBits/sec

out significant loss in performance. We use curves shown in Figure 10a. The performance degradation with message lost from the Feature Detector (FD) agent to the Feature Repository (FR) and messages lost from FR agents to Optimizer (Opt) agent is linear. Messages from Radars (Rad) to FR agents have some slack, which allows for a greater percentage of messages to be dropped before there is a significant effect on performance. Messages from Optimizers to Radars have almost no slack, and there is a significant drop in performance with drop in messages.

Since messages from FD to FR and Rad to FD have the same performance degradation, we gain no additional benefit by analyzing the two separately, and can perform better analysis by merging the role of FR and FD into a single role (calling it the FR/FD role). The priority is averaged and the modification is shown in Figure 10b

We use a sparse 16 agent network. The network is divided into 4 clusters. The density of the network is increased by keeping the area constant, but moving from 16 agents to 160 agents, placed randomly. An example 25 agent network is shown in Figure 11 (agents on the border between two clusters belong to the cluster where their center lies). For each cluster, we randomly assign the role of Opt to one agent and the role of FR/FD to another agent. The rest of the agents (unmarked in the figure) are all Radar agents. The Rad agent sends 100 1Kbit messages to the FD/FR agent every second. FR/FD agents send 25 1Kbit messages/second to the Opt agent and Opt agents send 5 1Kbit messages/second to the Rad agents.

In this experiment, we start with enough bandwidth to allow all messages to get through. Next, the bandwidth is reduced until a certain percentage of messages are dropped for each routing algorithm. For example, the threshold before 10% of messages are dropped is different for different routing algorithms. The effect of dropping messages, is calculated at different bandwidths. Figure 12 shows the performance degradation with messages loss. Note, the global performance across the three message types is calculated as the product[4] of individual performances.

Figure 12 shows that the global performance is much higher for the eCQRouting algorithm. This is because the low priority messages are dropped before high priority messages. OLSR, DSDV and CQRouting all performed the same when dropping the same percentage of messages, since the message loss is equivalent across all messages types for each of the three algorithms. Interestingly, no improvement was obtained using either one of the two priority techniques described in Section 4.2 (represented as eCQRouting with priority for the simple priority based measure and eCQRouting with performance measure for the message-loss based approach). This is primarily because both algorithms attempt to do the same optimization on the network. The differences arise in when the evaluation occurs and the degree of evaluation. With the priority based representation, the status of network and the goodness of the path are evaluated at the time of exploration. Exploration happens more frequently for source-destination pairs where the priority of messages is much higher, hence allowing them to find sub-optimal paths that provide better probabilities of message delivery. However, if the state of the network changes between exploration messages, the technique will have a delayed reaction to the change. On the other hand, the per-message decision is based on local knowledge of the next hop and expectation of the rest of the network. However the benefit of having global information is lost. In this experiment, it appears

[4]For simplicity, the performance at each agent is assumed to be independent of other agents, and hence multiplied to give us global performance

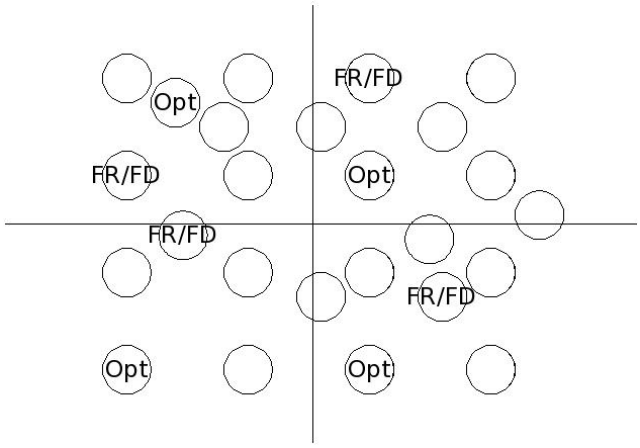


Figure 11: Example 25 agent network

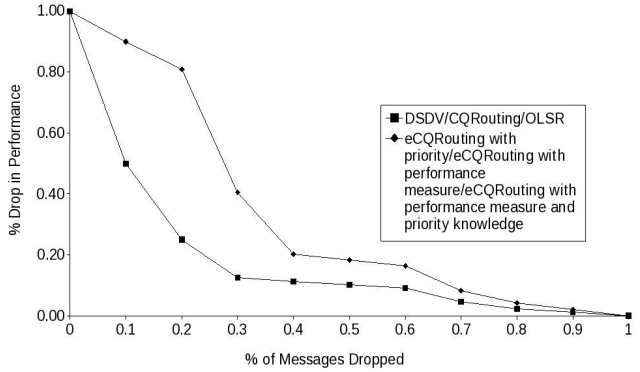
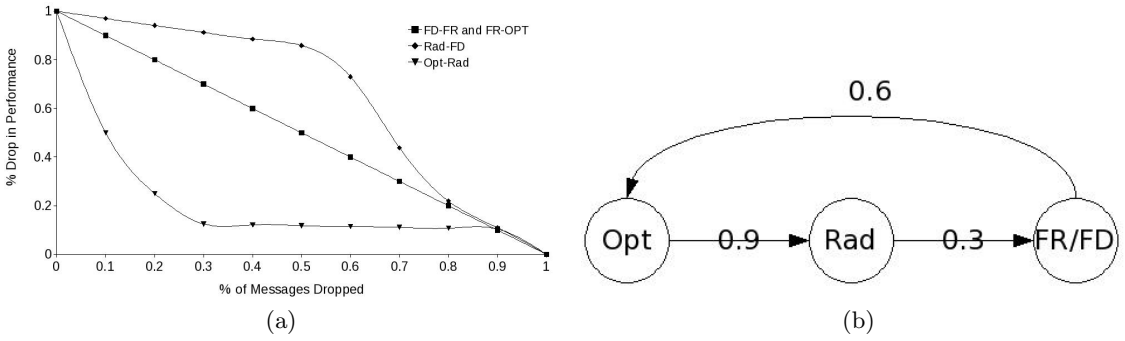


Figure 12: Performance degradation with loss of messages



(a) (b)

Figure 10: (a) Performance degradation as messages get dropped for messages from FD agents to FR agents, FR agents to Optimizer agents, and Optimizers agents to Radar agents. (b) Corresponding priority of messages

that the global and local benefit balance each other, resulting in similar results when used individually and a small 2% improvement when combined.

6. CONCLUSION AND FUTURE WORK

In this paper, we showed that using organization-level knowledge in network-level routing significantly improves application-level bandwidth relative to standard routing protocols. In a 100 agent simulation of the CNAS sensor network, we obtained an increase in available application bandwidth of 35% as compared to OLSR, one of the more widely used routing algorithm for wireless ad hoc networks. Additionally we observed an increase in global utility by focusing path exploration and, in turn, generating better paths for those source-destination pairs where communication has higher priority.

In this work, information flowed from the application-level organization to the lower-level routing algorithms. We assumed that the organizational structure was appropriate given the network structure. If we relax this assumption, an obvious extension of this work would be to allow the organizational structure to be modified based on the changing characteristics of the network. The network-level routing algorithm produces network capability and status information that can be used as input for possible adaptation or redesign of the organizational structure. Furthermore, as the organization structure changes, the updated information would be passed to the routing algorithm, increasing application bandwidth further. For example, cluster heads are picked dynamically in CNAS. However, we made the assumption that the cluster head was already chosen in the organizational knowledge provided to the routing algorithm. In practice, the routing layer could provide the sensor agent with a list of all other agents that it can currently communicate with that are within its cluster area. A decision of which of these agents is the cluster head could be made and then provided to the routing algorithm. Moreover, with the increasing research in emergent organizational design, a MAS application can start with no organizational structure, use the routing information to develop an initial organization design, which then provides exploration direction to the routing algorithm for better paths and bandwidth.

7. REFERENCES

[1] J. Boyan and M. Littman. Packet Routing in dynamically changing networks: A reinforcement learning approach. *Cowan, J.D.;Tesauro, G.;and Alspector, J., eds., Advances in Neural Information Processing Systems*, 1994.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol. *RFC 3626, Internet Engineering Task Force (IETF)*, October 2003.

[3] D. Corkill, D. Holzhauer, and W. Koziarz. Turn Off Your Radios! Environmental Monitoring Using Power-Constrained Sensor Agents. *First International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, 2007.

[4] D. Corkill and V. Lesser. The use of meta-level control for coordination in a distributed problem solving network. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, August 1983.

[5] B. Horling and V. Lesser. A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review*, 19(4):281–316, 2005.

[6] S. Keshav. REAL: A Network Simulator. *tech. report 88/472, University of California, Berkeley*, 1998.

[7] M. Krainin, B. An, and V. Lesser. An Application of Automated Negotiation to Distributed Task Allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, Fremont, California, November 2007.

[8] S. Kumar. Confidence based Dual Reinforcement Q-Routing: An On-line Adaptive Network Routing Algorithm. *Master's thesis, Department of Computer Sciences, The University of Texas at Austin, TX-78712, USA Tech. Report*, A:198–267, 1998.

[9] J. McQuillan and D. Walden. The ARPANET Design Decisions. *Computer Networks*, 1, August 1992.

[10] R. Onishi, S. Yamaguchi, H. Morino, H. Aida, and T. Saito. A multi-agent system for dynamic network routing. *IEICE Transactions of Communications*, 84-B(10):2721–2728, 2001.

[11] C. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, October 1994.

[12] C. Perkins and E. Royer. Ad-hoc On-Demand Distance Vector Routing. *Proc. 2nd IEEE Workshop. Mobile Comp. Sys. and Apps*, pages 90–100, 1999.

[13] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1989.

F Solar-Harvesting Model Development: The ATSN-08 Paper

This appendix contains a preprint of the ATSN-08 paper: "Simplifying Solar Harvesting Model-Development in Situated Agents using Pre-Deployment Learning and Information Sharing ," by Huzaifa Zafar and Daniel Corkill. This paper will appear in the *Proceedings of the Second International Workshop on Agent Technology for Sensor Networks* (ATSN-08), Estoril, Portugal in May 2008.

Simplifying solar harvesting model-development in situated agents using pre-deployment learning and information sharing

Huzaifa Zafar
Department of Computer Science
University of Massachusetts, Amherst
hzafar@cs.umass.edu

Daniel Corkill
Department of Computer Science
University of Massachusetts, Amherst
corkill@cs.umass.edu

ABSTRACT

Agents deployed in real-world applications typically need to determine various aspects of their local environments in order to make appropriate decisions. This must be done quickly, as performance can suffer until each agent develops a model of its environment. We use a strategy of factoring this model development (that would typically be done using multi-agent learning) into two phases: pre-deployment (site independent, individual agent) learning and post-deployment (site dependent, multi-agent) model completion. By performing as much learning as possible prior to deployment, we simplify what needs to be determined on-site by each agent. Furthermore, we use collective sharing of local-observation information, in conjunction with temporal and spatial constraints in relating information, to reduce the number of observations needed to perform each agent's model-completion activities. In this paper, we apply this two-phase strategy in developing prediction models for solar energy to be harvested by each agent in a power-aware wireless sensor network. In all but the most unlikely of environmental conditions, this strategy allows individual-agent harvesting models to be completed using only the first and second day observations.

1. ENVIRONMENTAL MODEL DEVELOPMENT

Multi-agent system (MAS) applications, such as wireless sensor networks (WSNs), suffer from performance degradation while agents develop models of their local environments. In developing these models, agents can either use observation information from other agents in the network (typically done using multi-agent learning (MAL)) or learn models individually (using traditional machine learning (ML) techniques). Agents using either MAL or ML require many observations before they are able to develop reasonable local models.

In order to develop these models faster, we introduce a strategy that factors model development into two phases; a pre-deployment learning phase, and a post-deployment model completion phase. The pre-deployment phase uses traditional ML techniques to develop a partial, parametrized, model of the environment that captures the agent-specific, site-independent aspects of the complete model. The site-specific factors are maintained as parameters in this model. By shifting all the site-independent learning to pre-deployment, an agent can dramatically reduce the information and time required to complete its model once situated.

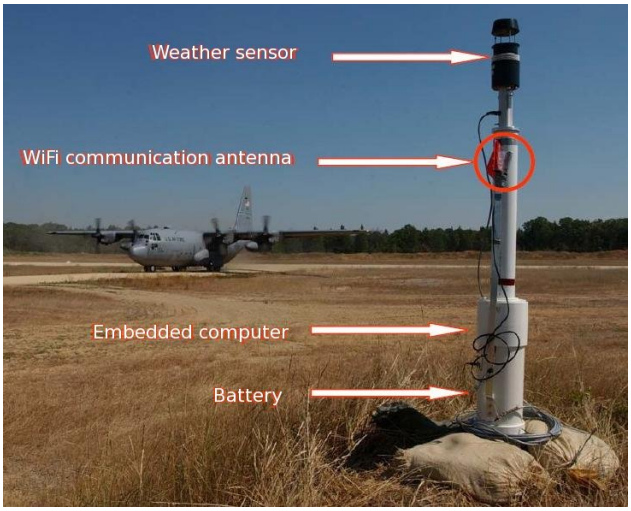


Figure 1: A TACMET-augmented CNAS sensor agent

Post-deployment model completion can be aided further by sharing information with other agents in the network. By using constraints defined on the parameters, an agent can use the shared information to constrain the site specific parameters in its local model. In fact, we show that, in all but the most unlikely of environmental conditions, an agent is able to develop an accurate solar harvesting model within two days of observations; but only if they use shared information from other agents in the network.

Similar separation strategies have been used in developing parametrized models with pre-deployment functional dependencies and then using those dependencies along with information sharing to improve the model post-deployment. Determining costs in network level routing algorithms is one such example [5, 7, 12]. However, the two-phase strategy presented in this paper is novel in shifting a majority (if not all) learning to the pre-deployment phase, and us-

This work is supported by the AFRL "Advanced Computing Architecture" program, under contract FA8750-05-1-0039. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views contained in this paper are the authors.'

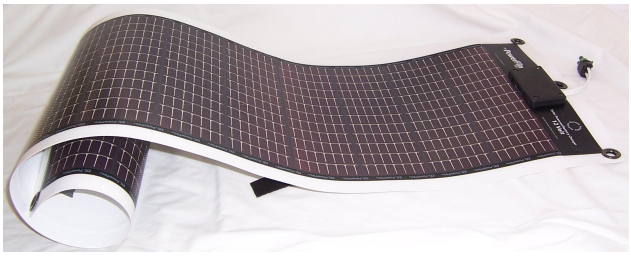


Figure 2: The rollable solar panel that we are using

ing post-deployment information sharing (along with constraints defined on parameter values) to complete a uniquely parametrized model with minimal local observations.

2. SOLAR HARVESTING MODELS

In this paper, we apply our two phase strategy to the development of solar-harvesting prediction models in the Collaborative Network for Atmospheric Sensing (CNAS) sensor network [3]. This model is used to predict the solar energy to be harvested by an agent given a cloudiness forecast for each time period in the following solar day. A TACMET-augmented CNAS sensor agent, shown in Figure 1, includes capabilities for weather sensing and WiFi based communication. Each agent is powered by an un-managed 12V battery (labeled in the figure). In order to extend the lifetime of the network, a rollable (thin film on plastic) solar panel (see Figure 2) is added to each agent. This allows the battery reserves to grow depending on the attenuation due to clouds, shade, and panel angle experienced by the agent during the course of the day. Moreover a requirement of CNAS is that the positioning of sensor agents cannot be optimized for either data collection or solar harvesting. Eventually sensor agents may be airdropped (rather than hastily deployed by hand). In either case, the exact placement of sensors cannot be regulated, which requires the shade and tilt attenuation to be determined once the agents are situated. Therefore, each agent in the network needs to develop the following environmental model:

$$E(t, l, \alpha) = E_{max}(t) * (1 - (f(C(t)) + g(S(t, l)) + h(T(t, \alpha))))$$

Where:

t = *time of day*

l = *geographic location of the agent (and its solar panel)*

α = *angle of the solar panel*

$C(t)$ = *function to determine % cloud cover given time*

$S(t, l)$ = *function to determine % shade cover given time and location*

$T(t, \alpha)$ = *function to determine % angle of solar tilt with respect to panel angle given time*

Figure 3 shows an example set of curves that illustrate the environmental model. In developing the solar harvesting model, we use an hour long time interval (using the averaged solar energy observed during the hour as our "observation"). However, any other time interval can also be used.

The observation by an agent ($E(t, l, \alpha)$) depends on the time period of the observation, the location of the agent,

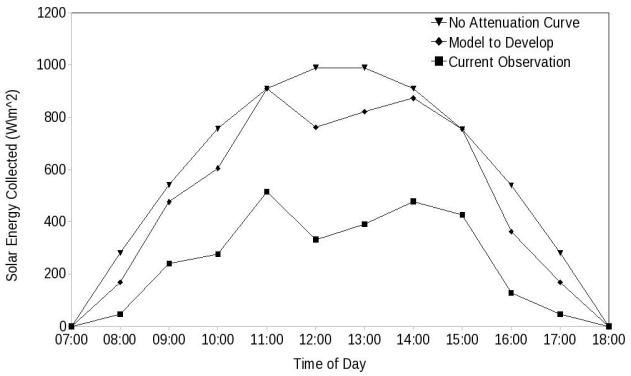


Figure 3: Example cumulative solar energy observed, along with a curve representing the local harvesting model of an agent

and the angle of its solar panel relative to the sun. This can be given by the unattenuated energy ($E_{max}(t)$) at time t for this agent's solar panel, reduced by the attenuation from clouds ($C(t)$), shade ($S(t, l)$) and tilt ($T(t, \alpha)$)[1]. The model also accommodates a non-linear effect of attenuation on the maximum energy observed. These relationships are given by functions f, g and h in the model.

We assume l and α to be constant for the lifetime of an agent (CNAS sensor agents are not mobile), and can be combined into a single parameter e that denotes the environment of the agent. The shade and tilt functions can be modified to $S(t, e)$ and $T(t, e)$. This makes both shade and tilt attenuation depend on the same parameters, allowing us to combine shading and tilting into a single attenuation called site attenuation ($S(t, e)$). Moreover, we can combine functions g and h, into a single function we call k, applied to $S(t, e)$.

3. PRE-DEPLOYMENT LEARNING

We start with the following environmental model (from the previous section) to be developed by each agent:

$$E(t, e) = E_{max}(t) * (1 - (f(C(t)) + k(S(t, e)))) \quad (1)$$

Once an agent develops a complete model, $C(t)$ will be provided for, and the agent is expected to use the rest of the parameters from Equation 1 in predicting an observation. Therefore the post-learning model is as follows:

$$E(t, e, C) = E_{max}(t) * (1 - (f(C) + k(S(t, e)))) \quad (2)$$

In moving from Equation 1 to Equation 2, the agent needs to retain functions E_{max}, f, k and $S(t)$.

The function E_{max} is dependent on the time of observation but not the site of the agent. Therefore, we can learn $E_{max}(t)$ pre-deployment. An example pre-deployment curve for $E_{max}(t)$ learned by an agent is shown in Figure 3 as "No attenuation curve".

[1]The accurate solar-harvesting model is more complicated than this, since the shade function will be applied to the residual energy after clouds have taken their effect. Similarly tilt applies after clouds and shade have taken their effect. However, we use the simple linear version to ease illustration. The reasoning remains the same for the more complicated model.

Similarly, $C(t)$ cannot be determined pre-deployment, since it depends on the cloud cover experienced by the agent at time t. However, the function f can be learned pre-deployment in the same way as E_{max}. An example curve is given in Figure 4.

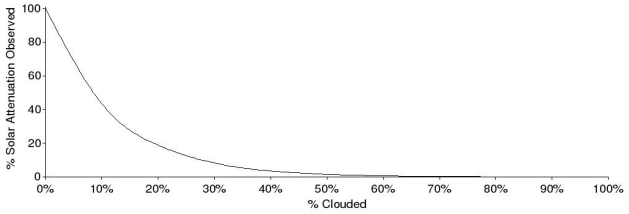


Figure 4: Relationship between cloud levels and solar energy harvested at noon, given no site attenuation

Finally, we can learn k by synthesizing various tilt and shade environments pre-deployment, leaving only the $S(t, e)$ function to be determined on-site.

Constraints

We make use of the following constraints on variables in our model;

1. For any given time in a day, the shading and tilting attenuation for an agent is constant across multiple days.[2]
2. We assume the cloud cover is spatially consistent enough that every agent experiences the same degree of cloud attenuation during any energy measurement period.
3. We assume that the effect of shading and cloud attenuation is such that, unless an agent experiences 100% cloud and site attenuation, it observes some solar energy value (possibly very small) that can be used to determine the degree of shading and clouding. Such non-zero observations are termed *meaningful*.

Defining Convergence

An agent's solar harvesting model is said to converge when, the agent can determine the value returned by the function $Site(t, e)$, given the constant e of the agent.

We define two levels of convergence. The first level is *hour*-convergence, where an agent converges (or is able to determine the value of S) for a given hour of the solar day. The second level is *day*-convergence, where the agent has obtained hour-convergence for every hour of the solar day.

4. SITUATED MODEL COMPLETION WITHOUT INFORMATION SHARING

We illustrate our strategy for completing the parametrized solar harvesting model (learned by each agent during the first phase) by looking at various examples, increasing in complexity. In the following three scenarios, we look at what a non-sharing agent can determine using only its local observations.

[2]We assume that the solar angle changes so little from day to day that it can be considered fixed. The model-completion phase of our strategy is fast enough that it can be repeated every few weeks to adjust for seasonal sun-angle change.

Scenario A

Description: No information sharing. No site or cloud attenuation.

Every observation the agent makes is at 100% of the expected maximum energy that can be collected for the hour. The agent can easily determine that the environmental attenuation is at 0% (that $E(t, e) = E_{max}(t)$). Once the agent determines its environmental attenuation, it achieves hour-convergence for that hour. Once the agent achieves hour-convergence for each hour in the solar day, it achieves day-convergence.

Scenario B

Description: No information sharing. No site attenuation. Varying cloud attenuation.

Since the observations the agent makes is less than its $E_{max}(t)$, the agent is unable to determine its environmental attenuation for the hour. For example, if an agent observes 50% energy loss, it can draw one of the three possible conclusions; 1) it lost 50% of its energy due to cloud cover, 2) it lost 50% of its energy due to site attenuation, or 3) it lost 50% of its energy due to some combination of cloud and site attenuation. Without additional information, the agent must assume anywhere between 0% to 50% of the energy is lost due to site attenuation. Since site and cloud attenuation are not necessarily constant across hours of the day, it has to make similar observations for all hours of the day. On subsequent days, if the agent makes observations greater than 50%, it can reduce the range of the expected loss due to site attenuation. However the only way an agent will achieve hour-convergence on its own is if it makes a 100% observation (meaning no site attenuation). Moreover, the agent will have to make such an observation for every hour of the day in order to achieve day-convergence.

In a later section, we will prove that an agent cannot determine the apportioning of site and cloud attenuation based on its own observations and requires observations from multiple agents within the network.[3]

Scenario C

Description: No information sharing. Varying site and cloud attenuation.

We assume that site attenuation is the same at the same time each day. However, site attenuation does vary across hours. This prevents an agent from achieving day-convergence by converging for a single hour. The scenario is similar to scenario B, and we can draw some of the same conclusions from it. If an agent makes a 50% observation, it will conclude site attenutation is responsible for 0% to 50% of the solar attenuation for the hour. However, since there exists some site attenuation, an agent will never make a 100% observation, and hence can never hour-converge on its own.[4]

5. SITUATED MODEL COMPLETION WITH INFORMATION SHARING

In the remaining 4 scenarios, we look at how information sharing improves convergence.

[3]This apportioning could be learned over many days given an expected cloudiness distribution. However, the number of days this would require precludes this approach.

[4](See previous footnote.)

Scenario D

Description: Observations from at least one hour-converged agent are received every hour. No site attenuation. Varying cloud attenuation.

We assume the hour-converged agent in this scenario can provide a meaningful observation. In the rest of the paper, as we talk about a day-converged (or hour-converged) agent sharing meaningful observations with other agents, we assume they can provide the degree of cloud attenuation experienced by themselves or other agents in the network.

Assume Agent 1 makes a 50% observation and goes through the same decision process as in Scenario B. Agent 2 also makes a 50% observation (since we assume identical cloud attenuation across agents). Since Agent 2 is hour-converged, it can share with Agent 1 exactly how clouded it is for this hour. This allows Agent 1 to determine conclusion 1 (no site attenuation) from scenario B and hence hour-converge, too.

If a single agent exists that has already hour-converged, all other agents in the network will hour-converge as soon they obtain a report from the converged agent. This is because the hour-converged agent is able to determine the degree of cloudiness experienced during the hour. As it shares this information with other agents in the network, agents are able to separate individual site and cloud attenuation, and they are able to determine site attenuation for that hour immediately, leading to their own hour-convergence.

Scenario E

Description: Information sharing among agents. No site attenuation. Varying cloud attenuation. No hour-converged agents.

Since cloud attenuation is constant at the same time across agents, all agents will make exactly the same observation regarding the effects of attenuation on their observations. Here sharing does not help, as neighboring agents are unable to provide any additional information because they cannot distinguish cloud and site attenuation. Similar to previous no-information-sharing scenarios, agents will have to experience a 100% observation to be able to hour-converge.

As shown in this scenario, if all agents make the exact same observation, no additional information is gained from sharing. Similarly, an agent can not learn anything new from hourly observations on corresponding days unless the observation has a higher energy value than a previous day's observation at the same hour. The analysis of this scenario highlights the convergence benefit of having site-attenuation variance among agents.

Scenario F

Description: Observations from at least one hour-converged agent are received every hour. Varying site and cloud attenuation.

This scenario is similar to scenario D. Since we have a hour-converged agent in our network, that agent is able to separate the site and cloud attenuation in its current observation. This allows it to determine the exact level of cloud attenuation and share this information with un-converged agents. Each un-converged agent uses the information to separate the two attenuation components from its own observation and hour-converge.

Scenario G

Description: Multiple information sharing agents. Varying cloud and site attenuation.

If all agents are site attenuated in exactly the same way (highly unlikely), this scenario becomes equivalent to Scenario E (as all agents make the same observation at each hour). Otherwise, there is at least one agent that makes a different observation, and all agents can gain information from this difference. For example, Agent 1 observes 70% of its maximum energy. For the same hour, Agent 2 observes 80%. Agent 1 can then conclude that it too should lose at most 80% of its energy due to cloud attenuation. This implies the remaining loss must be due to site attenuation, reducing the site attenuation range from 0%–30% to 10%–30%. Sharing therefore allows us to tighten the lower bound on the range, while the observations allow us to tighten the upper bound. Also, if Agent 3 makes a 100% observation, Agent 1 can immediately hour-converge by saying its losing 30% of its energy due to site attenuation. Finally if Agent 4 makes a 50% observation, Agent 1 can gain no additional information from Agent 4 since all 50% could be lost due to site attenuation. This shows us sharing is limited in the same way as hourly observations when assisting agents in developing a bound on the site attenuation.

A second benefit of sharing is hour-convergence without having to see a 100% observation. In fact, with sharing an agent can hour-converge, for any given hour, with 2 different observations for that hour, and 2 correspondingly different observations for the same hour by a different agent. The set of equations solved by the two agents over the two days is as shown below. For ease of expression, we assume $f(Cloud)$ and $k(S)$ are linear.

Agent 1:

Day 1:

$$1 - (Cloud(t_1) + Site(t_1, e_1)) = O(t_1, e_1) \tag{3}$$

Day 2:

$$1 - (Cloud(t_2) + Site(t_2, e_1)) = O(t_2, e_1) \tag{4}$$

Where $O(t, e) = \frac{E(t,e)}{E_{max}(t)}$. Also, constraints define $t_1 = t_2$ for the $Site$ function, since the the hour of the day is the same for both days.

Agent 2:

Day 1:

$$1 - (Cloud(t_1) + Site(t_1, e_2)) = O(t_1, e_2) \tag{5}$$

Day 2:

$$1 - (Cloud(t_2) + Site(t_2, e_2)) = O(t_2, e_2) \tag{6}$$

As long as $O(t_1, e_1) \neq O(t_2, e_1) \neq O(t_1, e_2) \neq O(t_2, e_2)$, the above equations are solvable, since we have 4 variables and 4 equations. Therefore, except under very unlikely environmental conditions, agents are able to hour-converge for each hour throughout day 2. We have already shown that all other agents hour-converge if 1 agent hour-converges. Note that domain constraints play a big part in arriving at this conclusion. Each agent can make use of the constraint that its site attenuation is the same at the same hour each day and that all agents have the same cloud attenuation at the same time. This allows any single agent to solve the 4 equations from observation reports received from other agents in the network.

Note that, a single agent cannot hour-converge even if it makes 4 observations over a period of 4 days. This is because each additional observation adds one additional variable as illustrated below:

Day 1:

$$1 - (Cloud(t_1) + Site(t_1, e_1)) = O(t_1, e_1) \quad (7)$$

Day 2:

$$1 - (Cloud(t_2) + Site(t_2, e_1)) = O(t_2, e_1) \quad (8)$$

Day 3:

$$1 - (Cloud(t_3) + Site(t_3, e_1)) = O(t_3, e_1) \quad (9)$$

Day 4:

$$1 - (Cloud(t_4) + Site(t_4, e_1)) = O(t_4, e_1) \quad (10)$$

Here we have 4 equations and 5 variables, which is unsolvable. Similarly, we cannot take 4 differing observations from 4 separate agents for the same hour as we would again end up with 5 variables.

6. EXPERIMENTAL RESULTS

In the last few sections, we analyzed the benefit of our two phased model-development strategy using a number of scenario cases. We call this strategy PLASMA (Pre-deployment Learning And Situated Model-development in Agents). In this section, we evaluate PLASMA operationally in a simulated solar-harvesting environment.

All experiments were performed using a simulator developed in GBBopen. For our first experiment, we use a network of 6 agents. The simulation spans 3 days. Cloud attenuation decreases from 50% on day 1 to 30% on day 3, with 10% decrements at the end of the day. Site attenuation varies from 30% for Agent 1 to 80% for Agent 6 with 10% increments with each agent. We show a similar curve as our previous experiment for Agent 2 that experiences 40% site attenuation in Figure 5. In the figure we show 4 curves.

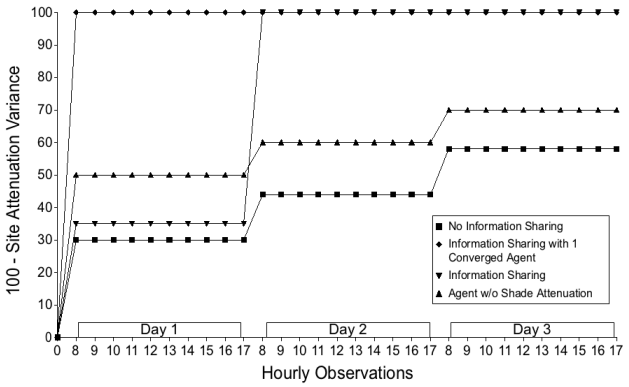


Figure 5: As the number of hourly observations increase, the range of the site attenuation determined by various scenarios in the model development strategy is explored

The first curve is a "No information sharing curve" wherein agents in our network do not share any information with one another. On day 1, An agent experiences 50% cloud attenuation and 40% site attenuation, for a combined 70% attenuation. Since it is not sharing any information, it determines its site attenuation to be anywhere in the 0%–70% range. Similar calculation is done for days 2 and 3. The second curve is "Information Sharing with 1 converged agent", wherein the agent shares information with another agent in the network that has already day-converged. The day-converged agent does not have 100% site attenuation and is able to predict the level of cloud attenuation for the hour, allowing the agent depicted in the graph to day-converge. The third curve is the "Information sharing" curve. With information sharing our agent is able to perform better on day 1, since it has another agent in its network which sees lower attenuation than itself, and is able to lower the range. However, on day 2, the agent is able to form 4 equations using observations from itself and its neighbor agents and day-converge. Finally for the fourth curve, "Agent w/o Site Attenuation" we remove site attenuation from the single agent. Also, no information sharing is allowed. This way, as the agent sees improving observations from one day to the next, it is able to improve its site attenuation range, but it can do no better without seeing 100% sun.

For our second experiment, we show the $E(t, l)$ function developed at the end of day 2, given 0% cloud attenuation. Here we have an network with 2 new agents. We showcase the curve learned by one of the two agents. The agent experiences clouds for the first 6 observations (first 6 hours of the day) and site attenuation for the remaining 4 hours. By using the 4 equations developed above, this agent is able to determine the degree of site attenuation experienced, and project the observations from the clouded periods to the expected maximum values. Figure 6 shows the corresponding model learned. In the figure, we showcase the benefit of both

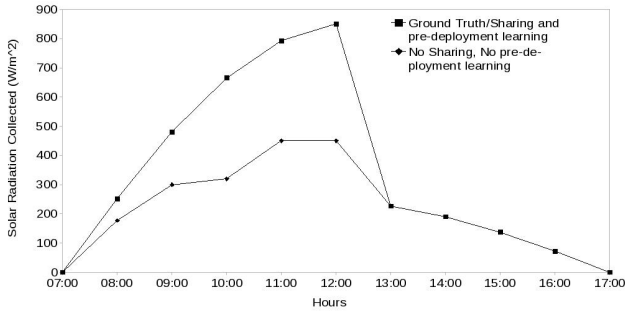


Figure 6: Model developed during the simulated run

sharing and pre-deployment learning. The ground truth is labeled as "Accurate Model", and represents what the agent should see given no cloud attenuation. Given observations with 100% sun for the first 5 hours, the agent is able to both learn the approximate attenuation due to clouds, and is able to use pre-deployed learning to project the current observation to the expected no cloud attenuation observation.

We are thus able to show, in our simulated environment, the benefit of both pre-deployment learning and collective information sharing demonstrated mathematically in previous sections.

7. POWER MANAGEMENT EXPERIMENTS

In this section we look at load assignment based on energy harvested by a group of agents in a wireless sensor network. This problem was looked at previously by Kansal et.al. [10]. We extend our simulation environment as follows: Each agent can undertake a certain task load. To accomplish this load, the agent requires a certain amount of energy. As the load increases, energy requirements also increase linearly. The agent earns a certain utility if a task is completed. The utility-load relation is the same as that used by Kansal et.al. and is shown in Figure 7.

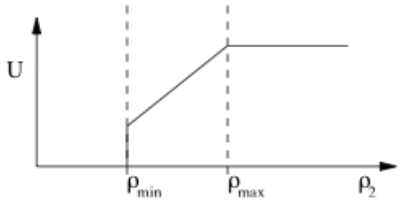


Figure 7: Utility and Task Size. ρ is the size of the task. Really small tasks have no utility. As the size of the task gets bigger than ρ_{min}, the agent starts increasingly gaining utility, which maxes out at ρ_{max}

We implemented the duty-cycle adaptation algorithm defined in Kansal et.al. [10]. In order to use their adaptation algorithm with the model developed by PLASMA, we modified the adaptation algorithm to make decisions solely on energy levels predicted for a given time period. Such a reactive algorithm could also be extended to take into account predictions of other agents within the system, which allows for better load balancing across multiple agents. However, this extension was not implemented so as to be able to compare results directly with Kansal et.al. We also implemented their Energy Prediction and Optimization model and compared it (using their original adaptation algorithm) with the local model developed by each situated agent in PLASMA (using the modified adaptation algorithm). For the experiments, we used a network of 10 agents. For each agent, we picked a random period of time during the day (of random size) to be shaded. Note, once the shading attenuation is randomly assigned to each agent, it remains fixed for all the remaining simulations as shown in Table 1. The simulation

Agent	Time period of shading	% Shade Attenuation
1	9:30 - 12:00	27%
2	15:30 - 16:30	80%
3	15:00 - 18:00	14%
4	10:00 - 11:30	32%
5	13:00 - 18:00	15%
6	13:15 - 14:00	66%
7	11:45 - 13:45	51%
8	10:30 - 13:00	42%
9	14:15 - 16:30	21%
10	9:00 - 10:00	37%

Table 1: Shade attenuation for agents in the simulation

now spans 20 days, with random cloud attenuation for every time period. A random cloud-attenuation pattern was established for the 20 simulated days and used in all simulation experiments. Mean cloud attenuation is around 25%, with about 20% variance. Each agent is asked to predict the expected solar energy for every time period, given expected cloud attenuation. The variance from the observation is calculated for each time period, for each agent, and averaged for each of the 20 days. Figure 8 compares the average variance calculated. As we can see from the figure, PLASMA

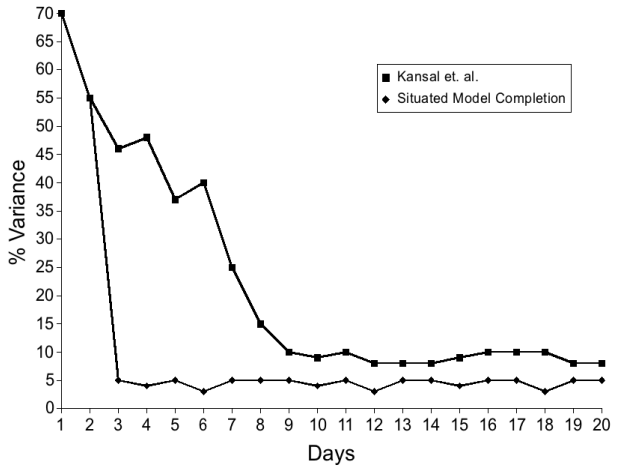


Figure 8: Comparing our strategy with the learning algorithm described by Kansal et.al.

is able to develop a more accurate model much faster than the Energy Prediction model defined by Kansal et.al. (the small variance is primarily due to the injected error in the cloud attenuation prediction provided to the agents, where mean error is about 5% with a variance of about 2%. This is to model the real world, where it is impossible to accurately predict cloud attenuation for any given time period. Also, since Kansal et.al. combine their learning algorithm with their prediction model (which does not take into account shade attenuation), their algorithm is unable to converge to PLASMA.

We compared utilities generated by the two algorithms in our 10 agent simulation. For the first experiment, each agent is provided with no accompanying battery, which forces each agent to make decisions based completely on the expected solar energy for each time period. We set ρ_{min} to 2 and ρ_{max} to 5. Also, agents can earn an utility between 0 and 5 every hour, depending on the amount of solar energy it can harvest for that hour and the task load acquired for the hour. An agents is penalized 1 utility point for each task it cannot complete. Figure 9 shows the results of the experiment. As we can see from the figure, PLASMA is fully informed starting on day 3 and achives maximum utility, except for days when the energy harvested is less than the energy required to perform all the tasks.

However, since Kansal's algorithm depends on some battery for their reactive algorithm, we provide each agent with a battery of infinite size to see how our strategy compares. Figure 10 shows the results. Here we see the Energy Prediction Model is able to do as well as our strategy once it converges to the correct learned values. The variance in learned values depicted in Figure 8 is covered by the infi-

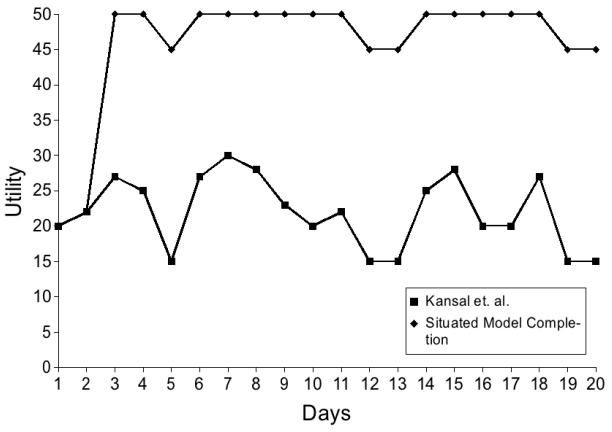


Figure 9: Utility is calculated by averaging the total utility earned by each of the 10 agents per day. This experiment assumes each agent has no battery attached.

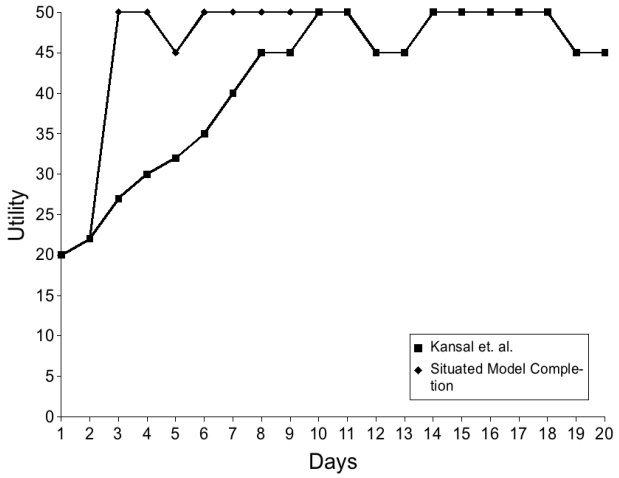


Figure 10: Utility is calculated by averaging the total utility earned by each of the 10 agents per day. This experiment assumes each agent has a battery of infinite size.

nite battery capacity, which allows Kansal et.al. to use all the energy harvested during the day. However, a lot of utility is wasted while the agent is trying to learn during the first 10 days. Also, beyond the first 10 days, how well the Kansal agent actually performs given a limited battery capacity will vary somewhere between performance given no spare battery (Figure 9) and performance given an infinite battery (Figure 10). The performance depends on how much of the difference between the prediction and actual solar energy harvested is buffered by the battery.

From the above three simulations, we can see the benefit of PLASMA both in the speed of convergence and lower battery buffer requirements.

8. APPLICATION TO OTHER DOMAINS

In order to be able to apply PLASMA to a model development problem, the domain must satisfy the following conditions:

1. **Pre-deployment and Post-deployment Phase**. A partial local model that captures the agent-specific, site-independent model is developed pre-deployment. Factors that are site-dependent remain as parameters within the parametrized model. Parts of the parametrized model that are dependent on the environment of the agent must be developed post-deployment. In the solar harvesting domain, the maximum energy curve of the solar panel and dependencies between the maximum energy and cloud and site attenuation were developed pre-deployment. Individual site attenuation was determined post-deployment to complete the model.

2. **Cooperative Information Sharing**. Agents exchange useful site-specific information to collectively improve local model completion. In the solar harvesting domain, agents share observations as well as cloud cover estimates from developed models in order to help other agents develop their own local models.

3. **Constraints**. Agents require constrains to relate observations made over time as well as observations made at the same time by other agents. In the solar harvesting domain, the parametrized model had two environmental factors, cloud and site attenuation. In order to converge, we required two constraints and 4 observations that related those factors using both constraints. If we increase the number of factors to three, we require 3 constraints and 9 observations to converge.

9. RELATED WORK

Multi-agent learning (MAL) is traditionally subdivided into multi-agent reinforcement learning (MARL) and multi-agent inductive learning (MAIL). MARL extends traditional reinforcement learning to multi-agent systems [2, 6, 13]. The disadvantage of MARL is the large observation set required for convergence. MAIL involves learning models by interacting with other agents in the network. Work in cooperative MAIL [1, 4, 9] involves determining what information needs to be shared between agents in the network and how to learn the entire model post-deployment. The disadvantages are similar to MARL in that many observations are required to develop a usable on-site model.

We applied our two-phase multi-agent model development strategy to solar harvesting models in WSNs. Sensor networks now incorporate agents that can harvest energy from their environment [8, 10, 11]. Energy-usage protocols have been developed that depend on energy harvested by individual agents in the network. These protocols are used in routing [16], duty cycling [15], and sleep wake cycling [14]. In each of these protocols, there is an added benefit in having each agent estimate the amount of energy it can harvest at any particular time based on cloud forecast and sharing this information with other agents in the network.

10. CONCLUSIONS

We presented a two-phase development strategy called PLASMA (Pre-deployment Learning And Situated Model-development in Agents) for MAS environmental models that uses a pre-deployment (agent-specific, site-independent)

phase and post-deployment (multi-agent, site-dependent) phase. By developing the majority of the model pre-deployment, we are able to simplify what needs to be accomplished once the agent is situated. Furthermore, we reduce the number of observations required by an agent post-deployment, by sharing information among agents. Finally, in order to relate observations from other agents to the local model, we take advantage of domain-specific constraints between the various environmental parameters in the model. The number of constraints and number of cooperative agents required to converge is related to the number of environmental parameters. Thus by developing a majority of the model pre-deployment, we can converge in a very short time.

In building its solar harvesting model, each agent is able to obtain day-convergence given observations from day one and two (except under extremely unlikely environmental conditions) when sharing information among agents. Hour-convergence requires: 1) another hour-converged agent that is able to provide a meaningful report or 2) four observations, over two days with different cloud attenuation across days and different shade attenuation across agents.

We also showed: 1) if a single agent within the multi-agent system day-converges, all other agents will also day-converge as soon as the day-converged agent shares its information with the rest of the network and the other agents have made observations throughout the day and 2) a non-interacting agent requires a single observation with no site or cloud attenuation in order to be able to hour-converge.

11. REFERENCES

[1] S. Benson. Inductive learning of reactive action models. In *International Conference on Machine Learning*, pages 47–54, 1995.

[2] P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde. Learning in distributed systems and multi-agent environments. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, pages 412–423, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[3] D. Corkill, D. Holzhauer, and W. Koziarz. Turn Off Your Radios! Environmental Monitoring Using Power-Constrained Sensor Agents. *First International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, 2007.

[4] W. Davies and P. Edwards. The communication of inductive inferences. In G. Weiß, editor, *ECAI '96: Selected papers from the Workshop on Distributed Artificial Intelligence Meets Machine Learning, Learning in Multi-Agent Environments*, pages 223–241. Springer-Verlag, 1997.

[5] Y. T. Hou, Y. Shi, and H. D. Sherali. Rate allocation in wireless sensor networks with network lifetime requirement. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 67–77, New York, NY, USA, 2004. ACM.

[6] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[7] Z. Hu and B. Li. On the fundamental capacity and lifetime limits of energy-constrained wireless sensor networks. *Real-Time and Embedded Technology and Applications Symposium. Proceedings. RTAS 2004. 10th IEEE*, pages 2–9, 25-28 May 2004.

[8] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *IEEE Information Processing in Sensor Networks*, pages 463–468, 2005.

[9] R. A. Jr., W. L. Johnson, J. Rickel, and A. Scholer. Learning domain knowledge for teaching procedural skills. In *AAMAS'02*, 2002.

[10] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. In *ACM Transactions on Embedded Computing Systems*, 2006.

[11] A. Kansal and M. Srivastava. An environmental energy harvesting framework for sensor networks. In *ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2003.

[12] L. L. L, N. Shroff, and R. Srikant. Asymptotically optimal power-aware routing for multihop wireless networks with renewable energy sources. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2:1262–1272 vol. 2, 13-17 March 2005.

[13] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[14] D. Niyato, E. Hossain, and A. Fallahi. Sleep and Wakeup Strategies in Solar-Powered Wireless Sensor/Mesh Networks: Performance Analysis and Optimization. In *Transactions on Mobile Computing*, pages 221–236, Feb 2007.

[15] C. Vigorito, D. Ganesan, and A. Barto. Adaptive Control for Duty-Cycling in Energy Harvesting-based Wireless Sensor Networks. In *Proceedings of the Fourth Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON 2007)*, San Diego, CA, June 2007.

[16] T. Voigt, H. Ritter, and J. Schiller. Utilizing solar power in wireless sensor networks. In *28th Annual IEEE International Conference on Local Computer Networks*, pages 416–422, October 2003.